

Учреждение образования  
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ  
УНИВЕРСИТЕТ»

**Кишкурно Т. В.**

# **ДИЗАЙН И ЮЗАБИЛИТИ ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЯ**

Тексты лекций по одноименной дисциплине для студентов  
специальности 1-40 05 01-03 «Информационные системы и  
технологии» (издательско-полиграфический комплекс)

Минск 2017

УДК 004.5(075.8)  
ББК 32.97я73  
К46

Рассмотрены и рекомендованы к изданию редакционно-издательским советом Белорусского государственного технологического университета.

Т. В.Кишкурно

Рецензенты:

заведующий кафедрой программного обеспечения информационных технологий БГУИР, доцент, кандидат технических наук

Н. В.Лапицкая

доцент кафедры логистики и информационно-математических дисциплин БИП• Л. И. Крошинская,

**Кишкурно, Т. В.**

К46      Дизайн и юзабилити интерфейсов пользователя: тексты лекций по одноименной дисциплине для студентов специальности 1-40 05 01-03 «Информационные системы и технологии» (издательско-полиграфический комплекс) / Т. В. Кишкурно. – Минск: БГТУ, 2017. – 155 с.

Тексты лекций содержат сведения о принципах создания удобных и привлекательных с точки зрения пользователя интерфейсов, о требованиях, предъявляемых к пользователе-ориентированному дизайну интерфейсов, о стадиях их проектирования и критериях качества. Рассмотрены основные способы прототипирования для реализации простых и сложных схем взаимодействия с пользователем и инструменты прототипирования, а также средства реализации на практике концепции юзабилити-тестирования, стратегию и варианты тестирования.

УДК 004.5(075.8)  
ББК 32.97я73

®УО «Белорусский государственный  
технологический университет», 2017  
®Кишкурно Т. В., 2017

## ПРЕДИСЛОВИЕ

Данный конспект лекций содержит материал предназначенный для студентов, обучающихся по направлению специальности 1-40 05 01-03 Информационные системы и технологии (издательско-полиграфический комплекс) при изучении дисциплины «Дизайн и юзабилити интерфейсов пользователя».

В настоящем конспекте, состоящем из восьми лекций представлены принципы создания удобных и привлекательных с точки зрения пользователя интерфейсов, требования, предъявляемые к пользователю-ориентированному дизайну интерфейсов, стадии их проектирования, стандарты по юзабилити и дизайну, а также критерии качества интерфейсов. Рассмотрены основные способы прототипирования для реализации простых и сложных схем взаимодействия с пользователем и инструменты прототипирования, а также средства реализации на практике концепции юзабилити-тестирования, стратегию и варианты тестирования.

Данный конспект лекций дает будущему специалисту широкий набор практических навыков по определению цели, задач и этапов проектирования интерфейсов, что позволит в дальнейшем эффективно использовать полученные знания в практической работе.

Сведения, получаемые при изучении данного курса будут также востребованы при изучении специальных дисциплин и станут инструментом для грамотного выполнения и оформления интерфейсов программных продуктов различной направленности.

Данный конспект лекций является полезным и при выполнении студентами лабораторных и курсовых работ по одноименной дисциплине.

## ЛЕКЦИЯ № 1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Наш предмет называется – «Дизайн и юзабилити интерфейсов пользователя». Прежде чем начать его изучение давайте определимся с понятиями, которые мы будем использовать.

### Пользовательский интерфейс

**Интерфейс пользователя** – (*UI – англ. user interface – пользовательский интерфейс*) – представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными устройствами и аппаратурой.

Интерфейс – только половина во взаимодействии с системой, другая половина – человек, пользователь. Чтобы интерфейс работал хорошо, нужно точно знать, что именно в любой конкретный момент пользователь: воспринимает в интерфейсе, думает, хочет добиться.

Интерфейсы являются основой взаимодействия всех современных информационных систем. Если интерфейс какого-либо объекта (персонального компьютера, программы, функции) не изменяется (стабилен, стандартизирован), это даёт возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами.

Например, научившись работать с одной программой Microsoft Office, пользователь с легкостью освоит и другие – потому, что они имеют одинаковый интерфейс.

Пользовательский интерфейс (ПИ) делиться на:

- **Интерфейс командной строки** – инструкции компьютеру даются путём ввода с клавиатуры текстовых строк (команд). В этом виде интерфейса человек подает "команды" компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки.

- **Графический интерфейс пользователя** (или **WIMP-интерфейс**: *Window* – окно, *Image* – образ, *Menu* – меню, *Pointer* – указатель) – программные функции представляются графическими элементами экрана. Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов – меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредственно", через графические образы. Этот вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и "чистый" WIMP – интерфейс.

• **Естественно-языковой интерфейс** (или **SILK-интерфейс**: *Speech* – речь, *Image* – образ, *Language* – язык, *Knowledge* – знание) – пользователь «разговаривает» с программой на родном ему языке. Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный «разговор» человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяли вначале в основном для военных целей.

В настоящее время наиболее актуально изучение технологии WIMP-интерфейса, так как он наиболее часто используемый для всех типов компьютеров.

## Дизайн

Понятие «дизайн» многозначно. Инженеры разрабатывают дизайн мебели, домов, новых видов материалов. Это слово используется при оформлении интерьера, одежды, ландшафта. Одни дизайнеры и конструкторы больше внимания уделяют внешнему виду, другие заботятся о цене товара и его продажи, третьи о соответствии дизайна потребностям пользователя. И это далеко не все факторы, которые учитываются в процессе разработки дизайна. Ведь конечный продукт должен удовлетворять всем заведомо противоречивым требованиям.

Дизайн – это сознательные и интуитивные усилия по созданию значимого порядка. Дизайн интерфейса позволяет придать определенный вид уже существующему поведению системы.

Программисты делают выводы: *«Я могу писать, как мне угодно, потому что «интерфейс» появится уже после того, как я закончу»*. Проектирование откладывается до завершения программирования, когда уже слишком поздно.

Дизайн интерфейса должен начинаться с людей. Вы разрабатываете для их потребностей, моделей поведения и желания. Вы будете смотреть на мир сквозь эту призму "привлекательности" в течение всего срока проекта. После того, как вы знаете, что желательно, вы начнете просматривать решения через призму «Технологии» и «Бизнеса». Дизайн, ориентированный на человека должен попасть в перекрытие трех линз (рис. 1): решения, которые привлекательны для людей, возможны и жизнеспособны.

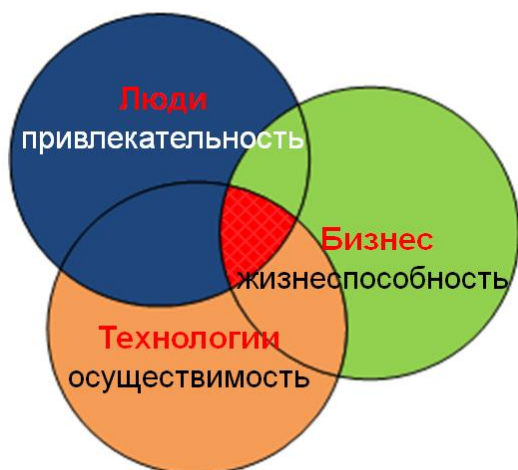


Рисунок 1

Он позволяет создавать решения, соответствующие потребностям и целям пользователей с одной стороны, а также бизнес-требованиям и технологическим ограничениям – с другой.

## Юзабилити

Согласно статистическим данным, большинство запросов в Google в начале нашего века по слову «usability» приходило из таких стран, как Индия, Малайзия, Сингапур, Новая Зеландия и Гонконг. России и Беларуси среди них не было. Интересно, что список этот во многом совпадает со списком стран, лидирующих в software-разработках. Получается, что разработчики из Индии интересуются юзабилити. Интересно, зачем?

Раньше представители IT-индустрии соревновались между собой за производительность «железа». Затем акцент переключился на функциональность «софта». Теперь конкуренция сместилась в область пользовательских качеств – юзабилити. Это происходит из-за того, что рынок технологий и программного обеспечения становится все более «массовым». Компьютеры, доступ в интернет есть практически в каждом доме. Мобильные телефоны, интернет-магазины, социальные сети, платежные терминалы – этими и другими программными продуктами ежедневно пользуются граждане, подавляющее большинство которых не являются профессионалами IT-индустрии.

И это не только активная молодежь. Среди «новых» пользователей – люди среднего возраста, выросшие без компьютеров, пенсионеры, не привыкшие обращаться с техникой, инвалиды и даже дети. Успех программного продукта в большей степени обеспечивается для них за счет его пользовательских качеств – того, насколько удобно его использовать, насколько понятен его интерфейс. Ориентация на пользователей становится крайне важной, чтобы выжить в этой конкуренции.

Не стоит думать, что это новый термин, пришедший к нам с Запада. В русском языке было очень похожее слово: эргономика, которая означало удобство использования предмета с целью минимизации ошибок. Возможно, вы слышали фразы “эргономичность станка” (станок, на котором возможность ошибки работником сведено к минимуму). Даже сейчас в рекламе проскакивает «эргономичное кресло» – т.е. настолько удобно сконструировано, что на нем можно жить.

Единственное более менее ощутимое различие между эргономикой и юзабилити – первое ставит больший акцент на технические характеристики, второе же – на процесс взаимодействия. В основе эргономики стоит сам продукт, в основе юзабилити – пользователь.

Наиболее часто используемое определение юзабилити (из стандарта ISO 9241–11) гласит, что: юзабилити – степень эффективности, продуктивности и удовлетворенности, с которыми продукт может быть использован определенными пользователями в определенном контексте использования для достижения определенных целей.

При разработке пользовательских интерфейсов словом юзабилити обозначают общую концепцию их удобства при использовании программного обеспечения, логичность и простоту в расположении элементов управления.

На самом деле не бывает абсолютной вещи удобной для всех. Она удобна сейчас для решения конкретной задачи конкретным человеком. Но как только здесь изменится какая-то переменная, то есть другой человек, не на которого рассчитывали. Например, рассчитывали на мужчин, а пришла блондинка или наоборот, сделали для блондинки, а пришел инженер и не соответствует продукт его ментальным моделям. Он пытается не по назначению использовать продукт. Например, сделали веб-сайт, а он его пытается с КПК или мобильного телефона смотреть. Но он не рассчитан на это.

Юзабилисты достигают золотой середины. Чтобы быть хорошим юзабилистом, необходимо быть хорошим психологом, чего не хватает всегда людям техническим. Они думают о технологии, а не о том с какой блондинкой встретится этот сайт. И как он должен ее завлекать.

Юзабилити – это технология влияющая на все этапы разработки программной системы. Основывается она на особенностях психологического восприятия информации человеческим мозгом. Основная её цель – проконтролировать, спрогнозировать и воздействовать на процессы создания системы с целью повысить конечную эргономичность продукта.

В процессе всего периода разработки интерфейса важно помнить, что именно так, как пользователь видит интерфейс на экране, так он и вос-



принимает весь продукт в целом. Следовательно, в его понимании, если программа неудобна, значит, она бесполезна во всех своих проявлениях. Тем более это относится к сайтам и играм – конкуренция в этом секторе рынка слишком высока, чтобы позволить себе создавать неудобные продукты. Почему это важно и зачем это нужно знать разработчику.

Юзабилити и дизайн важен каждому человеку причастному к разработке. Почему? Потому, что целью практически любого разработчика является привлечение потенциального пользователя. Пользователю же в свою очередь важно получать наиболее позитивные чувства. Его не интересуют внутренние процессы, он их не видит.

Есть программы, базы данных, сервера, но пользователь работает с интерфейсом. Интерфейс – это то, что видит пользователь, когда он работает с программой. Это наподобие айсберга. Человек видит только надводную часть, т. е. интерфейс. А значит, взаимодействуем мы с ним через интерфейс, который должен подчиняться определенным нормам. Для управления этими нормами и нужно юзабилити. Дизайн же нужен, для того чтобы все это грамотно воплотить в жизнь. В итоге наша основная цель, доставить пользователю наиболее приятные ощущения при взаимном обмене информацией, во время работы с нашей системой.

В тоже время юзабилити – это Usability Engineering – научно-прикладная дисциплина, служащая повышению эффективности, продуктивности и удобству использования инструментов деятельности, в том числе – программного обеспечения.

Научным фундаментом для юзабилити являются:

- **HCI** (*human–computer interaction, человеко–компьютерное взаимодействие*). Это научная и прикладная дисциплина, предметом которой является то, как люди используют компьютеры и как следует разрабатывать компьютерные системы, для того, чтобы обеспечить более эффективное их использование конечными пользователями. Дисциплина включает элементы психологии, эргономики, информатики, графического дизайна, социологии и антропологии.

- **UCD** (*user–center design, проектирование, ориентированное на нужды пользователей*). Подход к проектированию компьютерных систем (приложений, веб-сайтов), при котором во главу угла ставятся пользователи, их нужды и требования. При использовании данного подхода достигается высокий уровень юзабилити, что сказывается, например, на увеличении продаж (в электронном магазине).

- **Human-centred design: ISO 9241–210 [7]** – это существующий стандарт проектирования. Human-centred design учитывает не только



процесс взаимодействия пользователя с системой, но и контекст, т.е. каким образом человек будет с ней взаимодействовать (например, взаимодействие с сайтом родителей школьника, который пользуется сайтом и др.)

- **IA** (*information architecture, информационная архитектура*). Совокупность методов и приемов организации информации для облегчения выполнения людьми их информационных нужд по поиску и просмотру информации.

### **Система международных стандартов графических пользовательских интерфейсов**

Существует несколько общепризнанных стандартов по юзабилити и доступности, которые применяются в проектировании интерфейсов. Некоторые из них носят рекомендательный характер, в то время как другие являются обязательными. При этом характер требований может зависеть от страны, для которой создается продукт или от типа системы. Например, компания, которая разрабатывает корпоративную систему, предъявляет к своему продукту более жесткие требования, нежели веб-студия, которая проектирует электронный магазин. Компании, создающие ПО и веб-сайты для широкой публики абсолютно обязаны соблюдать стандарты.

Деятельность в области графических систем возглавляется и координируется Международной организацией по стандартизации (ISO – International Standardization Organization) при активном участии NIST (National Institute of Standards and Technology – Национальный институт стандартов и технологий США), IEEE (Institute of Electrical and Electronics Engineers – международная некоммерческая ассоциация специалистов в области техники, мировой лидер в области разработки стандартов по радиоэлектронике, электротехнике и аппаратному обеспечению вычислительных систем и сетей) и ряда крупных программистских фирм. Это неправительственная сеть национальных институтов по стандартизации, в которой участвуют представители из 146 стран. Стандарты, которые выпускает эта организация, формируют своеобразный технологический словарь, при помощи которого, производители продуктов могут разговаривать друг с другом на одном языке.

Существует несколько стандартов, разработанных в ISO и относящихся к сфере юзабилити и человеко-компьютерного взаимодействия:

- **ISO 9241** – содержит требования к эргономике визуальных дисплейных терминалов для офисной работы. Основной акцент сделан на требования к офисному оборудованию: дисплеям, клавиатурам, к от-

ражению, к цвету, к компоновке элементов на экране, к диалогам и сообщениям об ошибках. Этот стандарт не применим к проектированию интерфейсов для мобильных устройств. В 11-м параграфе дается определение юзабилити – «эффективность, продуктивность и удовлетворение пользователя», а также приводятся различные метрики, которые помогают проектировать удобные пользовательские интерфейсы.

- **ISO 13407** – описан процесс проектирования интерактивных систем, ориентированных на пользователей. Содержит рекомендации по организации процесса проектирования интерфейсов и органичному встраиванию этого процесса в общий процесс производства ПО. В стандарте описаны методы юзабилити, необходимые для: определения контекста использования продукта, выявления требований пользователей и заказчиков к системе, прототипирования и юзабилити-тестирования продукта.

- **ISO 18529** – эргономика человеко-компьютерного взаимодействия – описание процесса проектирования интерфейсов, ориентированных на пользователей. В стандарте детально описана модель зрелости организации с точки зрения уровня использования в ней UCD-процесса. Даются рекомендации по переходу на более высокие уровни зрелости.

- **ISO 14915** – эргономика программного обеспечения мультимедийных пользовательских интерфейсов. В стандарте даются рекомендации по созданию элементов управления для мультимедийных продуктов, таких, например, как обучающих систем, справочных киосков, электронных справок.

- **ISO 16982** – эргономика взаимодействия человек–система. Методы, основанные на удобстве применения, для обеспечения проектирования, ориентированного на человека.

Кроме этого в различных странах (США, Великобритании и т. д.) существуют законы, регламентирующие права людей с ограниченными возможностями. Например: сайты федеральных учреждений должны быть доступны всем, в том числе и людям с ограниченными возможностями. То есть те компании, которые заявляют, что следуют стандарту должны: понимать и учитывать контекст использования (включая характеристики пользователей, задач и среды), направлять процесс проектирования в соответствие с пользовательскими требованиями, создавать решения, которые соответствуют этим требованиям, проводить оценку решений с привлечением пользователей с изменением решений в соответствие с выявленными недостатками.

## ЛЕКЦИЯ № 2. ПСИХОФИЗИЧЕСКИЕ ОСОБЕННОСТИ ВОСПРИЯТИЯ ИНФОРМАЦИИ ЧЕЛОВЕКОМ

Создаете вы сайт, медицинское оборудование или любой другой продукт – ваша целевая аудитория состоит из людей и должны хорошо знать:

- Как люди думают, как они принимают решения.
- Что заставляет человека нажать на кнопку или купить что-нибудь.
- Как заставить людей сделать то, что вы хотите.
- Какие ошибки они делают и почему.
- Как привлечь их внимание.

Любой интерфейс должен учитывать особенности человеческой психологии, физические ограничения человека и его сознания во время работы, способность человека отвлекаться во время работы, неточность его движений, восприятие того что он видит, физическое напряжение во время работы.

### Восприятие информации

Разные люди имеют разные психофизические особенности. Человек имеет пять видов чувств, с помощью которых он осваивает внешний мир. Это – зрение, слух, вкус, обоняние, осязание.

Процесс приёма и усвоения информации, получаемой при помощи воздействия предметов и явлений на органы чувств человека, называется **восприятие** или **перцепция** (*от лат. perceptio*). Этот процесс построен на взаимосвязи психических и биологических функций организма.

Общеизвестно, что 90 % информации об окружающем мире человеку дает зрение, 8 % приходится на органы слуха, обонянию и того меньше. Зрение является главным каналом восприятия информации.

Во время прогулки или на экскурсии наши глаза воспринимают информацию и передают ее в мозг, который обрабатывает ее и представляет реалистичную картину того, что нас окружает. Но наши глаза не работают так, как фотоаппарат, объективно фиксирующий мир. Они действуют совместно с мозгом, который определенным образом «истолковывает» видимый мир. Изображение, попадая в мозг, изменяется и интерпретируется. Наш мозг является хранилищем, в котором находятся миллионы «образцов» объектов, и когда мы видим объект, мы сопоставляем его с содержащимися в памяти «образцами».

Представление не несет ничего нового, так как объект восприятия и представления один и тот же. Более того, при восприятии объект обрисован точнее, нежели чем при представлении. Но преимущества представления есть, и их нужно искать в ином.

**Восприятие** – это актуальные переживания, то есть те, которые воздействует на человека именно в этот момент. Следовательно, восприятие происходит до тех пор, пока какое-либо воздействие продолжается. Другое дело **представление**. Оно воспроизводит то, что действовало на человека когда-то, то есть было отражено в виде восприятия.

Отличительный момент в том, что в представлении мы можем пережить то, что совсем не связано с актуальной ситуацией (пример на рис. 2). Стало быть, оно позволяет заново пережить те ощущения, которые были пережиты в иное время.



Рисунок 2

На успешное восприятие визуальной информации влияют такие факторы как:

- **Размер объекта** (большие объекты воспринимаются острее); Если два аналогичных объекта разного размера показать одновременно, то меньший объект будет восприниматься как расположенный дальше (от наблюдателя), чем более крупный объект. Размер знакомых объектов также можно использовать для указания размеров и глубины незнакомых объектов. Для печатного текста наиболее оптимальным считается шрифт в 9 – 12 пунктов. Крупный шрифт используется для аудитории, состоящей из лиц пожилого возраста.

- **Цвет объекта** (в некоторых пропорциях цвет способен создавать конкретное настроение и привлекать к себе внимание; и соответственно наоборот);

- **Интенсивность объекта** (в эмоциональном или физическом значениях);

- **Контрастность объекта** (выделение объекта из окружающей среды). Использование темного текста на светлом фоне и наоборот. Изображение оптимально в том случае, когда уровень контраста между текстом и фоном составляет 70%.

Цвету присуще ассоциативное восприятие и определенная информационная нагрузка.

Вы когда-нибудь замечали, что предприятия быстрого питания пользуются одной палитрой? Преобладают красный и желтый оттенки, некоторые бренды, которые не используют эти цвета, замещают их оранжевым, розовым и другими теплыми, яркими цветами. Все потому, что они следуют Теории Цвета. Она гласит, что некоторые цвета обладают способностью вызывать определенные чувства, эмоции или поведение людей. Если вы знаете эти скрытые свойства, то сможете использовать их в своем интерфейсе для достижения желаемого результата, как это делают предприятия быстрого питания

При создании интерфейсов дизайнеры используют Теорию Цвета для улучшения эстетики, читабельности страниц, и, что более важно, для получения нужного психологического воздействия.

Вместо того чтобы использовать цвета, которые просто будут неплохо смотреться, вы можете основать свой дизайн на получении реального эффекта – используя опыт исследований воздействия цвета на наше мышление. Опираясь на Теорию Цвета, вы сможете улучшить эстетику, читабельность страниц, и, что более важно, достигнуть максимального психологического воздействия, которое позволит привлечь пользователей и свести к минимуму вероятность того, что они не заинтересуются вашим интерфейсом.

Однако для разных людей цвета могут иметь различные значения, и многое зависит от культуры страны, в которой они проживают. Например, белый цвет в США, да и у нас символизирует чистоту, в него надеваются невесты. А в Индии – цвет траура и похорон.

Подбирайте цвета внимательно, учитывая те ассоциации, которые они могут вызвать. Выберите несколько главных культур или стран, для которых будет предназначаться ваш дизайн, и проверьте коррект-

ность использования цветов по цветовой карте культур на <http://www.informationisbeautiful.net/visualizations/colours-in-cultures/>

Все цвета спектра уникально красивы, но при этом каждый из них по-своему влияет на человеческое настроение и эмоции (рис. 3).

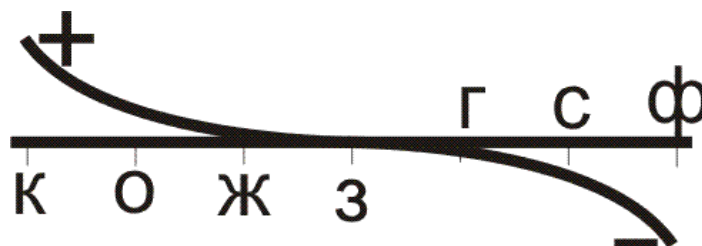


Рисунок 3

Пояснения к рисунку: + тонизирующее воздействие, – успокаивающее.

Например, красный – возбуждающий, согревающий, активный и энергичный цвет, проникает и активизирует все функции организма. Интуитивный, цвет физической силы. Это сильный цвет, и вы должны знать о некоторых его негативных эмоциях: опасность, тревога. Например, красный для финансиста означает влезть в долги или потерю денег, также он может означать сигнал опасность или знак «стоп».

Зеленый, как видно из рисунка 3 занимает самое нейтральное положение. Он воздействует противоположно красному. Зеленый успокаивающий, естественный цвет, способный ассоциировать ваш интерфейс с чувством безопасности и спокойствия. Это цвет гармонии, природы, исцеления, жизни, питания и здоровья. Зеленый – отличный выбор, если вы хотите вселить чувство доверия в своих посетителей. Зеленый ассоциируется также с деньгами (по понятным причинам). Зеленый – прибыль или «проходите».

Фиолетовый цвет, на самом деле, должен находиться как бы в стороне, т.к. он имеет угнетающее воздействие, а не успокаивающее как у синего или голубого цвета.

### Когнетика

Когнетика учитывает статистическую природу различий между людьми. Изучение прикладной сферы наших ментальных способностей называется когнитивным проектированием или **когнетикой** (от слова *Cognate* – родственный, сходный).



Для описания аспектов функционирования нашего мышления в психологии, философии и истории применяются термины **сознательное и бессознательное**.

При проектировании интерфейса пользователя имеет смысл использовать более ограниченные понятия **когнитивное сознательное и когнитивное бессознательное**.

**Когнитивное бессознательное** – это те ментальные процессы, которые вы не осознаете в тот момент, когда они происходят.

**Когнитивное сознательное** включается в тот момент, когда вы сталкиваетесь с ситуацией, которая кажется новой или представляет угрозу или когда требуется принять нешаблонное решение.

Свойства когнитивного сознательного и когнитивного бессознательного сведены в таблицу 1.

Таблица 1.

Свойство	Сознательное	Бессознательное
Иницируется	Чем-то новым, нестандартными ситуациями, опасностью	Повторением, ожидаемыми событиями, без-опасность
Используется	В новых обстоятельствах	В привычных ситуациях
Решает задачи	Принятия решений	Работа с неветвящимися задачами
Принимает	Логические утверждения	Логические или противоречивые утверждения
Функционирует	Последовательно	Одновременно
Управляется	Волей	Привычными действиями
Производительность	Небольшая	Огромная
Период функционирования	Десятки секунд	Десятилетия (всю жизнь)

Интерфейс должен снижать когнитивную нагрузку на пользователя. Она должна быть минимальной. Здесь, прежде всего, следует учитывать нагрузку на основные психические процессы: *память, внимание, воображение*.

**Когнитивная нагрузка** относится к общему количеству информации, с которой может справиться память человека. Когнитивная перегрузка происходит, когда память получает больше информации, чем она может свободно обработать, что приводит к проблемам при принятии решений.



## **Внимание**

**Внимание** – это то, что позволяет нам обрабатывать информацию об окружающем нас мире. Мы осознаем вещи только тогда, когда внимаем им, обращаем на них свое внимание. Внимание часто сравнивают с фонариком, который мы направляем на объекты для того, чтобы выделить их из бесконечного множества других предметов. После того как мы сфокусировали луч от фонарика на этом объекте, мы тут же начинаем осознавать и интерпретировать то, что предстало нашему взору.

Человеческий мозг обладает некоторыми особенностями восприятия окружающей нас действительности, которые приносят разнообразие нашему мышлению и развитию. Превосходя по всем параметрам и возможностям любой компьютер, наш мозг отличается от машины тем, что умеет сконцентрироваться на определенном объекте и вычитать из своего внимания не интересующие.

**Фокус внимания** человека применительно к компьютерным системам – некоторое место на экране, куда направлен его взгляд и где он сознательно сосредоточен. Фокус внимания может быть только один. В любой момент времени человек может сосредоточить свое внимание только на одном предмете. Это может быть какой-то объект реального мира (например, лист бумаги) определенная область экрана или окна, а может и какой-нибудь процесс «в уме» (например, когда человек обдумывает свои действия или что-то рассчитывает). Предмет, на котором сосредоточено внимание человека, называется **локусом его внимания**.

**Локус внимания** – это некоторое место или область, на которое может быть сосредоточено ваше внимание. В отличие от фокуса, часто обозначающего не только место, но и действие (сфокусировать ваше внимание), локус обозначает только место и переводится с латинского, как место положения или область. Мы можем целенаправленно сфокусировать наше внимание на каком-либо локусе.

Видимый предмет не всегда может быть локусом вашего внимания. Локус внимания может быть только один. Информация, ставшая локусом внимания, перемещается в кратковременную память, где храниться в течение 10 секунд.

С локусом внимания связано как минимум две особенности человеческого восприятия. При смене локуса теряется связанная с ним «оперативная» информация, которая содержится в кратковременной памяти. Соответственно, при возвращении к прежнему локусу эту ин-

формацию необходимо каким-то образом восстанавливать. Например, при периодическом переключении внимания, с рабочей области документа на уведомления об ошибках, эффективность работы снижается.

При пристальном сосредоточении внимания все события вне локуса могут игнорироваться или просто оставаться незамеченными.

Мы можем сами управлять своим вниманием. Такой вид внимания называется **произвольным (намеренным)**. Например, когда мы находимся на лекции, мы заставляем себя слушать речь лектора и смотреть на доску или экран. Или когда мы ищем, какую-то книжку в библиотеке, мы сознательно обращаем внимание на названия книг, написанных на корешках.

Сигналы из внешнего мира, другие объекты, которые находятся в поле нашего восприятия, также могут управлять нашим вниманием. Например, громкий хлопок вынуждает нас повернуть голову вслед предполагаемому источнику звука, а мигающая иконка в углу экрана моментально приковывает наш взгляд. Этот вид внимания является **спонтанным** и он управляется внешними стимулами. Психологи называют такой вид внимания **непроизвольным**.

Внимание очень сильно влияет на производительность труда, особенно того труда, который связан со взаимодействием человека и компьютера. Интерфейс ПО или веб-сайта **должен управлять вниманием пользователя**, помогая тем самым воспринимать ту информацию, которая является значимой «здесь и сейчас».

Существует три вида внимания:

**1. Избирательное внимание.** Этот вид внимания иногда называют туннельным вниманием. Оно возникает тогда, когда мы обращаем внимание на стимул или задачу так страстно, что начинаем полностью игнорировать все остальные стимулы и объекты. Программист, занятый написанием кода программы, геймер, бороздящий просторы виртуального пространства, или водитель, полностью сконцентрированный на дороге, все они могут запросто пропустить мимо ушей вопрос, заданный им другим человеком.

При поиске в «зашумленном» интерфейсе некоторой важной информации мы также используем свое избирательное внимание.

**2. Фокусированное внимание.** Это внимание можно назвать более эффективным избирательным вниманием, потому что в данном случае мы целенаправленно перестаем обращать наше внимание на стимулы для того, чтобы завершить задачу. Например, пользователь программы, который сознательно игнорирует уведомление о новом сообщении, мерцающем в углу экрана, для того, чтобы закончить и

послать e-mail. В поле фокусированного внимания находится письмо, а остальные стимулы человек намеренно исключил из своего поля зрения. Например, если пользователь пытается сохранить в MS Word документ с именем уже существующего файла, то выводится модальное предупреждение (рис. 4), которое привлекает внимание пользователя и не позволяет ему отвлекаться на что-либо еще.

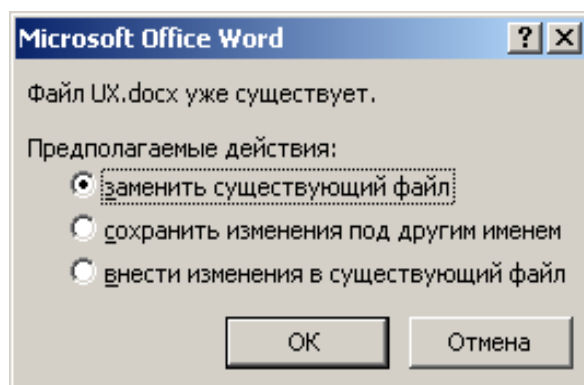


Рисунок 4

**3. Распределенное внимание.** Бывают такие ситуации, когда становится невозможным фокусировать свое внимание на одной задаче из-за того, что другие стимулы начинают отвлекать нас. Например, если мы вдруг услышим, как кто-то разговаривает о нас в то время, пока мы ведем беседу с другими людьми, нам станет трудно удерживать свое внимание исключительно на нашем собственном разговоре. Некоторые компьютерные задачи вынуждают пользователей фокусироваться на нескольких вещах. Это значительно снижает эффективность и продуктивность работы пользователя.

Распределенное внимание является особо критическим фактором для задач, при выполнении которых требуется особая бдительность (то есть те задачи, где пользователь должен отслеживать изменения в интерфейсе в течение длительного времени). Примеры таких задач: контроль над воздушным трафиком, над системой безопасности, управление процессом на атомной электростанции.

### **Центральное и периферийное зрение**

Помимо типов внимания важно то, куда направлено визуальное внимание в каждый данный момент времени. Существует два типа зрения:

**Центральное зрение** — обеспечивается центральным участком сетчатки и центральной ямкой. Дает человеку возможность различать

формы и мелкие детали предметов, поэтому его второе название – форменное зрение.

**Периферийное зрение** – обеспечивает ориентацию человека в пространстве, дает возможность видеть во тьме и полутьме. Кроме объекта, на который вы смотрите, в поле зрения попадает также большое количество различных вещей. Вы видите все эти предметы нечетко, но, все же, видите, имеете возможность улавливать их движение и реагировать на него.

Когда пользователь использует избирательное или сфокусированное визуальное внимание, тогда он использует центральную область сетчатки глаза (**центральное зрение**) с самой большой концентрацией фоторецепторов или другими словами ту область интерфейса, которая находится в пределах всего поля зрения пользователя. Эту область часто называют UFOV (*useful field of view* – *полезное поле зрения*). UFOV обычно находится между 1 и 4 градусами угла зрения и это факт нужно учитывать в задачах, где информация вне этой области может быть пропущена.

Наше периферийное зрение охватывает зону приблизительно в 207 градусов. Мы используем наши уши, чтобы следить за звуками и определять, когда нам нужно повернуть голову, чтобы получить зону видимости в 360 градусов. У нас есть зона высокого разрешения (центральное зрение), размер которой составляет немного меньше, чем размер листа бумаги, находящегося на расстоянии, обычно используемом для чтения.

Наше периферийное зрение использует всю сетчатку, а зона высокого разрешения использует только область, которая имеет диаметр всего 0,2 миллиметра (рис. 5).

Интересно, но женщины и мужчины видят несколько по-разному. Из-за определенных различий в строении глаз представительницы прекрасного пола способны различать больше цветов и оттенков, нежели сильная часть человечества. Кроме того, ученые доказали, что у мужчин лучше развито центральное зрение, а у женщин – периферическое. Страховые компании отмечают, что женщины намного реже, нежели мужчины, попадают на автомобилях в аварии, которые связаны с боковыми ударами на перекрестках. Зато параллельная парковка дается прекрасным дамам сложнее.

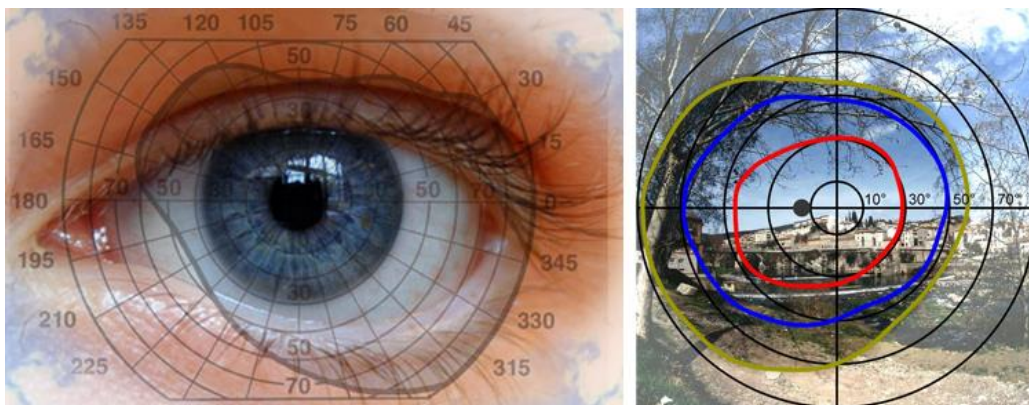


Рисунок 5

Также женщины лучше видят в темноте, в близком широком поле замечают больше мелких деталей, если сравнивать с мужчинами. В то же время, глаза мужчин хорошо приспособлены к слежению за объектом на дальнем расстоянии. У прекрасных дам глаза устают медленнее, нежели у мужчин.

Периферическое зрение человека независимо от его желания фиксирует движение. Например, если вы читаете текст, а на экране присутствует анимация или постоянное изменение яркости и цвета по краям экрана, вы не можете не обращать на это внимание. Если вам необходимо сосредоточиться на тексте, подобные ухищрения веб-дизайнеров могут достаточно сильно раздражать. Это работает периферическое зрение. Именно поэтому рекламщики используют изменение яркости и цвета в объявлениях, расположенных по краям веб-страниц. Это раздражает, но привлекает внимание.

Мы используем наше периферийное зрение для установки приоритетов информации, а наш мозг фильтрует и сортирует огромные объемы данных, уделяя внимание только тому, что определяет, как важное.

Проектировщики интерфейса должны понимать принципы работы периферийного зрения и визуальных подсказок, которые можно использовать для привлечения внимания к той области интерфейса, которая находится вне UFOV. Периферийная область нашего зрения самая чувствительная к визуальным подсказкам, основанным на движении, мигании и резким изменениям в контрасте.

### **Визуальные подсказки**

Понимание принципов и законов, по которым работает внимание, позволяет создавать эффективные пользовательские интерфейсы.



Интерфейс должен помогать пользователю, фокусировать внимание на важной информации. Можно заострить внимание на текущем объекте в системе с помощью визуальных подсказок. Например, **положения объекта**.

В 2006 году исследователь Якоб Нильсен обнаружил, что мы просматриваем веб-страницы по определенной схеме (рис. 6). Посетители сайта сканируют информацию на странице по F-образной форме, потому что взгляд пользователя обычно скользит слева направо в направлении к нижней части страницы. Зная, что пользователи будут просматривать веб-страницу по F-образной кривой и потратят на это несколько секунд (не больше 5-8), разместите по такой схеме основные элементы целевой страницы.

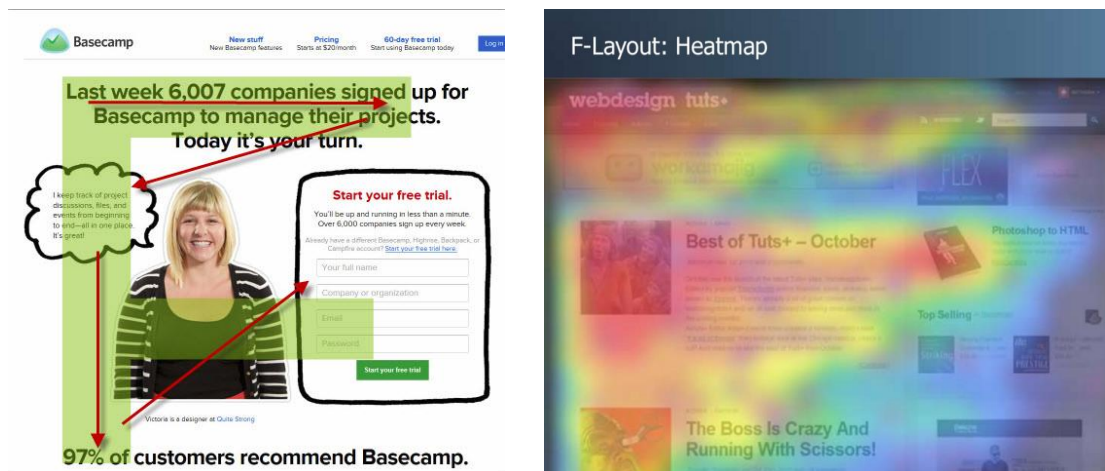


Рисунок 6

Размеры, цвет и контрастность — все эти свойства могут играть роль визуальной подсказки, которая привлечет внимание пользователя к чему-то существенному.

Например, использование синего цвета ссылок на веб-страницах является оптимальным т. к. этот цвет во-первых контрастный по отношению к цвету фона и не сливается с основным цветом текста и во-вторых у пользователей есть определенный опыт восприятия веб-страниц, они привыкли использовать его каждый раз, когда попадают на новую страницу или сайт. Можно использовать и другой цвет более привлекательный, по вашему мнению, но это может снизить скорость поиска, с которой посетитель будет находить ссылки.

Можно привлекать внимание пользователей и с помощью **подсветки**. Обычно система подсвечивает текущий объект. Т. е. при

наведении на них курсора их цвет меняется. Например, в браузере Internet Explorer последних версий панель инструментов представляет собой серые пиктограммы, которые становятся цветными при наведении на них мыши. В web частным случаем подсветки является изменение цвета и вида гиперссылки при наведении на нее мыши.

Можно **указать** какую-либо область экрана или объект, если это поддерживается программой. Например, может быть указана конкретная строка таблицы после «клика» на ней мышью. Указание – это вариант «долговременной» подсветки.

Для обозначения объекта, над которым нужно произвести некоторое действие используют **выделение**. Выделить объект можно, установив флажок в чек-боксе или нажав радио-кнопку. Можно выделить часть текста, изображение или какую-либо еще часть содержимого страницы.

Если мы хотим сделать объект системы доступным для использования или преобразования, то его необходимо **активировать**. Например, выпадающее меню в некоторых случаях может быть использовано только после активации управляющего элемента, т.е. наведения на него курсора. Текущий раздел меню, как правило, идентифицируется при помощи подсветки.

Кроме этого есть способы, позволяющие привлечь внимание пользователя к ссылкам или функциям, которые призваны обеспечивать цели бизнеса или самого пользователя. Платная регистрация на сайте знакомств – хороший пример функции, которая является ключевой для бизнес-цели. А ссылка «подробнее» – пример ссылки, которая помогает пользователю сориентироваться и получить полный текст статьи, то есть решить свою задачу.

Очень важно знать, какие элементы нужно сопровождать визуальной подсказкой, когда и каким образом. Например, навигационные ссылки и пункты меню должны всегда сопровождаться визуальными подсказками, иначе посетитель может пропустить целый раздел сайта. Кнопки, которые относятся к важным действиям (например, «зарегистрироваться» или «послать») должны не только быть похожими на кнопки, но и выделяться среди других элементов.

Кроме того, в опыте каждого пользователя есть целый набор элементов, которые он отфильтровывает, считая их незначимыми и не относящимися к делу. Пользователи уже научились игнорировать баннерную рекламу и избирательно обращать свое внимание лишь на те элементы и функции, которые кажутся им значимыми. Этот эффект получил название «**баннерная слепота**».



## Память

На качество взаимодействия пользователя с системой существенно влияют возможности человеческой памяти. На этот фактор мы не в состоянии повлиять. У каждого конкретного человека есть свои особенности памяти, т. е. хорошая у него память или плохая, какая память у него более развита (зрительная, слуховая, осязательная), какой объем его кратковременной памяти и т.д.

Для нас актуально знать про две подсистемы памяти, а именно про **кратковременную (КВП)** и **долговременную (ДВП)** подсистемы.

Вся информация, воспринимаемая пользователем при работе с системой, хранится в кратковременной памяти, которая является, по сути, не отдельным свойством человеческого мозга, а некоторой составляющей долговременной памяти. Как информация попадает в КВП?

Представьте себе коробку без боковых стенок, в которой один за другим лежат кирпичи. Если с одной стороны попытаться засунуть в эту коробку еще один кирпич, то он туда, безусловно, влезет, но при этом с противоположной стороны вывалится другой, оказавшийся лишним. Таким образом, в коробке всегда находится ограниченное число кирпичей. Так и человеческая память всегда сохраняет текущую информацию, важную в данный момент. Как только потребность в ней исчезает, на ее место приходит новая информация, которая «вытесняет» лишнее из области памяти.

Соответственно, чтобы что-либо попало в КВП пользователя, он должен это заметить (для чего, собственно говоря, и полезно проектировать интерфейс с учетом возможностей человеческого восприятия) и счесть полезным лично для себя. Таким образом, **самое важное в интерфейсе должно быть наиболее заметным** (вот мы и узнали теоретическое обоснование очередного очевидного факта).

Другая интересная особенность КВП заключается в том, что содержание в ней происходит при появлении новых стимулов. Практический смысл этого наблюдения состоит в том, что **нельзя допускать, чтобы пользователь отвлекался, поскольку новые стимулы при отвлечении стирают содержимое КВП**. Необходимо максимально облегчать возвращение пользователя к работе.

Кратковременная память имеет весьма ограниченный объем.

Считается, что человеческая память способна запомнить семь плюс-минус два элемента. Оценивать объем КВП применительно к интерфейсу как всеобъемлющие  $7 \pm 2$  элементов не вполне правомерно.

Во-первых, в КВП информация хранится преимущественно в звуковой форме. Это значит, что вместо смысла запоминаемых элементов в КВП хранится текст, написанный на этих элементах. Для нас это означает, что ***подвергать ограничению следует преимущественно те элементы, которые содержат текст.***

Во-вторых, известно, что в память помещается гораздо больше, но только в тех случаях, когда элементы сгруппированы. Соответственно, всегда ***можно сгруппировать элементы и поместить в КВП пользователя больше информации.***

Например, большинство людей за 30 секунд могут запомнить список из пяти слов. Но некоторые могут за это же время запомнить из 10-ти слов. Разбиение этого списка на более мелкие фрагменты (например, две или три группы) приравнивает такой процесс запоминания к списку из 5-ти слов.

В-третьих, существует некоторое количество людей, способных удержать девять значений в КВП, но количество людей, способных удержать в памяти только пять или шесть значений, тоже довольно существенно. Это значит, что с практической точки зрения гораздо удобнее считать, что объем КВП равен ровно семи элементам (или, если ситуация позволяет, шести), поскольку рассчитывать нужно не на сильное, а на слабое звено.

Так что значительно эффективнее считать, ***что объем кратковременной памяти равен пяти (шести, из которых один в запасе) элементам.*** Не более, но и не менее.

И запоминание, и извлечение информации из памяти требует усилий. Более того. Поскольку содержимое КВП теряется при поступлении новых стимулов, пользователям приходится прилагать усилия, чтобы просто удержать информацию в памяти (вспомните, сколько раз вы повторяли номер телефона, чтобы удержать его в памяти на время, пока вы переходите в другую комнату).

Таким образом, необходимо снижать нагрузку на память пользователей, т.е. избегать ситуаций, когда пользователю приходится получать информацию в одном месте, а использовать её в другом.

Объем ДВП очень велик, информация, попавшая в ДВП, хранится, судя по всему, вечно. Интерес представляют два вопроса: при каких условиях информация попадает в ДВП и сколько «стоит» воспоминание.

Оба вопроса очень интересны с точки зрения обучения пользователей, второй вопрос, к тому же, интересен еще и с точки зрения улучшения способности пользователей сохранять навыки работы с

системой в течение длительного времени (а это одна из основных характеристик хорошего интерфейса).

Считается, что информация попадает в ДВП в трех случаях.

Во-первых, при повторении, т.е. при зубрежке. Чем больше повторений, тем больше шансов, что информация будет запомнена. С точки зрения дизайна интерфейса это наблюдение вызывает очень простую эвристику: если системой придется пользоваться часто, пользователи ей обучатся, деваться-то им некуда. Это очень утешительное наблюдение.

Во-вторых, при глубокой семантической обработке (*Семантика – свойство, определяющее смысл информации как соответствие сигнала реальному миру*). Если пользователь долго мучается, стараясь понять, как работает система, он запомнит её надолго, если не навсегда. Чем больше человек думает о какой-либо информации, чем больше он соотносит её с другой информацией, уже находящейся в памяти, тем лучше он запомнит то, о чем думает. Несколько помогает понять устройство механизма запоминания его антипод, а именно забывание. Самое простое объяснение имеет затухание: когда информация не используется долгое время, она забывается.

В-третьих, при наличии сильного эмоционального шока. Эмоциональный шок нас интересует слабо – не стоять же, в самом деле, за спиной у пользователя, стреляя время от времени из ружья, чтобы он волновался (тем более что после шока запоминание прерывается). Достаточно и повторения с обработкой.

Таким образом, повторение можно охарактеризовать как способ мощный, но ненадежный, поскольку трудно рассчитывать на повторение при нечастой работе с системой (существует множество систем, используемых редко или даже однократно). Семантическая же обработка есть способ мощный, но дорогой: без повода пользователи не будут задействовать свой разум, предоставить же им повод сложно.

Лучше всего в качестве повода работает аналогия, неважно, как она представлена, как метафора интерфейса, или как эпитет в документации.

Считается, что обращение к ДВП стоит довольно дорого. Поспорить с этим невозможно, поскольку в утверждении содержится слово «довольно», обладающее крайне размытым значением.

На самом деле всё сложно. Разные понятия вспоминаются с разной скоростью, слова, например, вспоминаются быстрее цифр, а визуальные образы – быстрее слов.

Для того чтобы пользователь запомнил, как работать с системой, ему необходимо либо постоянно «загружать» в кратковременную па-

мать одинаковые фрагменты (повторять), либо сознательно запоминать возможные способы работы с системой (обучаться), либо получать постоянные консультации по ходу работы (подсказки и помощь). Так постепенно получаемая информация переходит из кратковременной памяти в долговременную. Быстрой обработке информации способствует такой интерфейс, который не требует серьезного обучения, обеспечивает поддержку пользователя, понятность и предсказуемость всей системы.

## **Привычки**

Интерфейс призван формировать у пользователя привычки. Привычки формируются независимо от того, думает ли об этом разработчик. Если одно и то же действие повторяется несколько раз подряд, оно становится привычным. Привычки могут возникать на умственном и на физическом уровне, постепенно переводя сознательные действия в бессознательные. Способность формировать привычки, если ее использовать правильно, может приносить положительные результаты. Например, унифицированный интерфейс большинства приложений MS Windows позволяет, освоив работу с одним приложением, например MS Word вполне успешно работать с MS Excel.

Однако, для того чтобы интерфейс формировал привычки, он должен соответствовать следующим требованиям:

- Интерфейс должен быть достаточно простым. Примитивно устроены современные кофемолки. Нажатием одной кнопки – прибор включается, и выключается. А в стиральной машине нельзя использовать метод «одного выключателя», как в кофемолке, но в то же время многие домохозяйки утверждают, что научить обращаться со стиральной машиной можно даже обезьяну.

- Интерфейс должен достаточно часто использоваться для формирования привычек. Многократно повторяемые действия ведут к автоматизму и созданию привычки, а перерыв в совершении этих действий ведет к некоторой потере контроля над ситуацией. Недаром пользователи, пусть даже опытные, по вопросам настройки компьютера обращаются к системным администраторам. Если не заниматься подобной настройкой очень часто, то некоторые тонкости забываются.

Привычки высвобождают внимание. Человек с высвобожденным вниманием устает меньше и меньше напрягается. Последствия привычек – так называемые предопределенные действия. Свойство интерфейса формировать привычки может приносить весьма положительные результаты и упрощать работу.

### ЛЕКЦИЯ № 3. ЭТАПЫ ПРОЕКТИРОВАНИЯ ИНТЕРФЕЙСА

Прежде чем мы будем говорить о проектировании, посмотрим, как развивался процесс разработки программного обеспечения в ИТ-индустрии согласно Алану Куперу (рис.7).

В ранние дни развития ИТ-индустрии процесс разработки сводился к тому, что программисты вынашивали идею продукта, а затем создавали и самостоятельно тестировали его.

В более поздние времена к процессу стали подключаться профессиональные управленцы, их задачи сводились к оценке потребностей рынка и формулированию основных требований к разрабатываемому программному обеспечению.

С развитием ИТ-индустрии в самостоятельную дисциплину выделилось тестирование, а также широкое распространение получили графические интерфейсы пользователя, появилась необходимость разработки различных визуальных элементов, в связи с чем, к процессу разработки ПО подключились графические дизайнеры и тестировщики.

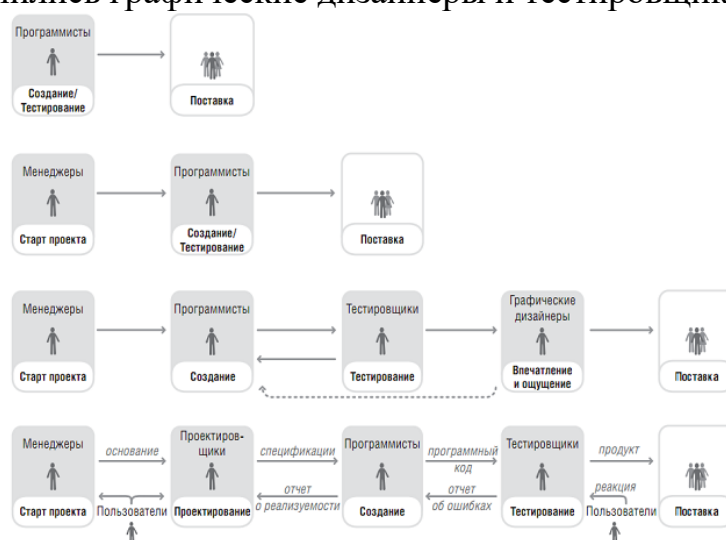


Рисунок 7

Сейчас используется подход к разработке программного обеспечения, при котором решения о возможностях продукта, его форме и поведении принимаются до начала дорогостоящей и сложной фазы создания продукта. Это обеспечивается включением в процесс разработки этапа проектирования, при этом эффективность проектирования во многом определяется выбранным стилем принятия решений, можно выделить 5 таких стилей:

- «непреднамеренное» проектирование – команда сосредоточена на разработке и внедрении приложения, не задумывается об удобстве его использования;

- проектирование «для себя» – основывается на опыте использования продукта членами команды, имеет большие шансы на успех, чем "непреднамеренное" проектирование, хорош, когда члены команды являются главными пользователями разрабатываемого продукта;

- Genius проектирование – основывается на опыте всех членов команды в проектировании подобных продуктов, хорошо работает, если уже есть опыт проведения предварительных исследований пользователей и сценариев их поведения с последующей проверкой соответствия дизайна ожиданиям пользователей;

- проектирование, ориентированное на деятельность – основывается на исследовании поведения пользователей, для исследования часто применяются методики, основанные на деятельности, например, построение диаграмм последовательности операций (work flow diagrams) и ориентированное на задачи тестирования удобства использования;

- проектирование, ориентированное на пользователя – основывается на глубоком исследовании целей и нужд пользователей, контекста использования, позволяет принимать детальные решения, которые были бы невозможны при использовании других методов.

Стили перечислены в порядке возрастания объема исследований, которые необходимо провести команде проектировщиков для принятия решений. На первый взгляд должно казаться, что оптимальным будет использование стиля, ориентированного на пользователей, но это не всегда так. Существуют проекты, для которых затраты времени и сил на проведение детального исследования пользователей не оправданы, с другой стороны существуют проекты, которые без таких исследований просто провалились бы. Умение для каждого проекта выбрать наиболее эффективный стиль проектирования является одной из ключевых характеристик хорошего проектировщика. Наиболее эффективные команды должны владеть всеми пятью методами принятия решений в процессе проектирования и выбирать именно тот, который лучше бы соответствовал нуждам и целям проекта.

## **Проектирование взаимодействия**

Для успешного проектирования цифровых интерактивных продуктов, к каковым относятся и большинство программных приложений, необходимо помнить о сложном поведении, которое эти продук-



ты демонстрируют. Очевидно, что необходимо особое внимание уделять проектированию поведения, при этом следует отметить, что традиционные дисциплины проектирования и дизайна нечасто затрагивают данный аспект. В связи с этим появляется новая дисциплина проектирования, которая получает название: проектирование взаимодействия и сосредотачивается в основном на проектировании поведения программного продукта.

Для решения этой задачи эффективно применяется инструментарий, включающий методику персонажей, текстовые сценарии взаимодействия, а также проектирование, ориентированное на цели. Весь этот набор можно объединить под названием: проектирование взаимодействия. При таком подходе к проектированию за отправную точку принимается человек, главная цель выяснить, чего хочет пользователь.

Проектирование взаимодействия предоставляет описание окончательного варианта продукта, которое содержит предельно ясную и точную информацию о том, кто конкретно будет использовать продукт, каким образом и с какой целью. Имея под рукой такое описание, программисты осознают, что именно они создают, руководители могут оценить прогресс в работе программистов, а маркетологи получают понимание источника мотивации покупателя.

Уже в начале нынешнего века стало понятно, что опыт взаимодействия при общении с цифровыми продуктами необходимо учитывать при разработке этих самых продуктов, пользователи устали от «технологии ради технологии». Крупнейшие учебные заведения, такие как, Стэнфордский университет и Harvard Business School признали, что следующее поколение управленцев и технологов должно находить проектированию взаимодействия место в своих бизнес планах и графиках разработок – и это необходимо учитывать в программах их подготовки. В сентябре 2005 года официально родилась организация IxDA – Interaction Design Association (ассоциация проектирования взаимодействия). Можно считать, что проектирование взаимодействия наконец становится самостоятельной дисциплиной и профессией.

Разумеется, проектирование продукта не может концентрироваться только на поведении, необходимо учитывать внешний вид (форму) и информационное наполнение (содержание) разрабатываемого продукта. Поэтому необходимо сочетать подходы различных дисциплин проектирования: проектирование взаимодействия, основное внимание на поведение; информационная архитектура, занимается структурированием содержания; промышленный и графический дизайн, отвечает за форму продуктов и услуг.



Этим всем занимается **UX дизайн (UX design) – User Experience Design** в переводе означает «опыт взаимодействия» и включает в себя различные UX-компоненты: информационную архитектуру, проектирование взаимодействия, графический дизайн и контент.

В целом, UX дизайн подразумевает комплексный подход к взаимодействию пользователя с интерфейсом, будь то веб-сайт, мобильное приложение или любая другая программа. Человек, который занимается этой работой – UX-дизайнер (в последнее время все чаще можно услышать названия UX-архитектора, UX-инженера или стратега, так как слово «дизайн» в данном контексте скорее нарицательное, чем то, что мы на самом деле привыкли понимать под значением этого слова) – при разработке интерфейса должен по возможности максимально учесть все мелочи, начиная от среды пользователя и типа электронного устройства и заканчивая способами ввода и отображения информации.

Простой пример: допустим, вы вложили внушительную сумму денег, чтобы продвинуть ваш ресурс на первые строчки поисковых систем, однако его удобство оставляет желать лучшего. В таком случае внушительное количество пользователей просто уйдет с сайта и эффект от этого будет минимальным. Именно поэтому необходимо проводить постоянный анализ действий посетителей ресурса, совершенствовать свой сайт и следить за современными тенденциями.

Основные вопросы, решаемые UX дизайном:

- Постановка целей и задач – чего в итоге нам необходимо достичь?
- Подбор подходящих UX инструментов для реализации целей.
- Разработка продукта, максимально удобного и легкого в восприятии целевой аудиторией
- Анализ конечного результата – соответствует ли продукт ожиданиям заказчика и насколько высок уровень удовлетворенности пользователей.

Именно грамотная продуманность всех деталей на этих этапах позволит создать армию поклонников вашего продукта. Ярким примером здесь является компания Apple, которая пошла по такому пути и завоевала сердца тысяч и миллионов.

Джесс Гаррет в своей книге «Веб-дизайн. Элементы опыта взаимодействия» представляет проектирование опыта взаимодействия в виде 5 уровней (рис. 8): стратегии (или идеи), набора возможностей, структуры, компоновки, поверхности (или внешнего вида интерфейса).

Эти пять уровней – составляют концептуальную основу для обсуждения связанных с опытом взаимодействия проблем и средств их решения. Пять наших слоев делятся на две части. Слева – все, что касается использования Web, как интерфейса программы. Справа – все, что связано с информационной структурой.

Каждый уровень зависит от уровней, расположенных ниже и диапазон выбора решений ограничен решениями, принятыми на нижних уровнях. На каждом из уровней размещаются элементы проектирования.

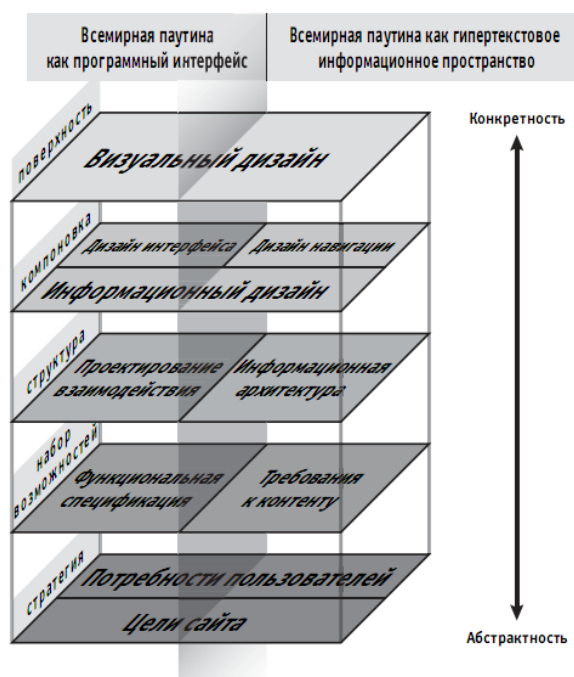


Рисунок 8.

#### Уровень стратегии:

**Цели сайта** (*Site Objectives*) – это бизнес-цели заказчика.

**Потребности пользователей** (*User Needs*). Они определяются людьми, которые будут пользоваться нашим проектом. Нужно понимать, чего хочет аудитория и как эти пожелания согласуются с другими ее потребностями.

#### Уровень набора возможностей:

**Функциональная спецификация** (*Functional Specification*) – подробное описание функций и возможностей, которые могут быть предоставлены пользователю.

**Требования к контенту** (*Content Requirements*) – это описание различных элементов содержимого, которые необходимо создать.

#### Уровень структуры:

Определяется, как подчинены друг другу разделы, какова иерархия построения информации, как осуществляются переходы между разделами. Если предыдущие уровни визуальные, то уровень структуры логический.

**Проектирование взаимодействия** (*Interaction Design, IxD*) – выяснение, как система будет вести себя в ответ на действия пользователей.

**Информационная архитектура** (*Information Architecture, IA*) – организацией элементов содержимого в пределах информационного пространства.

**Уровень компоновки:**

Это схематичное разделение страницы на части. Здесь нет информации о внешнем представлении – только о смысле конкретной части страницы, например, блок поиска, блок новостей, блок авторизации. На этом уровне определено, как должны располагаться части страницы.

**Информационный дизайн** (*Information Design*) – представлением информации в таком виде, который облегчает ее восприятие.

**Дизайн интерфейса** (*Interface Design*) – организация элементов интерфейса, позволяющая пользователям взаимодействовать с функциями системы.

**Дизайн навигации** (*Navigation Design*) – набор экранных элементов, позволяющих пользователю перемещаться по информационной архитектуре.

**Уровень поверхности:**

Это самый верхний, презентационный уровень. Он представляет собой совокупность шрифтов, таблиц, изображений, цветов, результатов поиска – всего, что видит пользователь. На этом уровне определено, как выглядит для посетителя содержимое сайта. Место для ввода текста.

**Визуальный дизайн** (*Visual Design*) – внешний вид конечного продукта. Для получения действительно привлекательных продуктов, необходимо, с одной стороны, понимать желания, потребности, мотивации пользователей и контекст, в котором эти пользователи находятся, с другой стороны, понимать возможности, требования и ограничения бизнеса, технологии и предметной области. Использование этих знаний в качестве основы всех планов по созданию цифровых продуктов позволяет сделать их полезными, удобными и желанными, а также экономически жизнеспособными и технически осуществимыми.

Три качества: желанность, жизнеспособность и осуществимость – определяют успешный продукт. Ларри Кили (Larry Keeley) выделил

современные задачи разработки, которые как раз предполагают реализацию в продукте всех трех качеств и, если хотя бы один из этих трех столпов значительно слабее двух других, продукт вряд ли выдержит испытание временем.

Если говорить об интерфейсе в целом, то сегодня не существует единой методологии их проектирования, поэтому мы будем использовать сочетание различных подходов.

## **Этапы работы над пользовательским интерфейсом**

В процессе разработки интерфейса можно выделить 5 основных этапов:

1. Сбор функциональных требований (первоначальное проектирование: уровень стратегии, уровень набора возможностей).

Данный этап разработки подразумевает под собой сбор, систематизацию и анализ требований к системе. Также анализируются и систематизируются возможные пользовательские системы (персонажи, актеры). Сбор и анализ требований выполняет бизнес-аналитик.

2. Информационная архитектура (уровень структуры).

Под информационной архитектурой понимается совокупность методов и приемов структурирования и организации информации. Другими словами, информационная архитектура занимается принципами систематизации, навигации и оптимизации информации, что позволяет облегчить пользователю работу с данными, а именно их поиск и обработку. За проектирование скелетов пользовательского интерфейса и организацию информационных потоков в приложении отвечает информационный архитектор.

3. Прототипирование пользовательского интерфейса

Этот этап разработки, который подразумевает создание прототипов экранов системы. Прототипы позволяют обнаружить проблемы функционального характера будущей системы на раннем этапе и устранить их до того, как проект уйдет в разработку к программистам. Прототипы разрабатываются front-end разработчиком под руководством UI-дизайнера.

4. Юзабилити – тестирование (Ю-тестирование).

К тестированию интерфейса привлекают как конечных пользователей, так и специалистов по функциональному тестированию ПО. Ю-тестирование позволяет оценить удобство использования продукта. Информационный архитектор проводит Ю-тестирование и анализирует его результаты.

5. Графический дизайн пользовательского интерфейса

Графический облик интерфейса создает UI-дизайнер. На этом этапе интерфейс системы приобретает необходимый законченный вид. Часто заказчик уже имеет брэнд бук или гайдлайн, задача дизайнера – разработать такой дизайн, который бы соответствовал всем требованиям системы, удовлетворял бы заказчика и сочетался бы с задумками информационного архитектора.

На данном этапе может понадобиться помощь смежных специалистов: иллюстратора (художника), 3D-моделлера и других.

**UI дизайн (UI design)** – User Interface Design или пользовательский интерфейс – это более узкое понятие, включающее в себя определенный набор графически оформленных технических элементов (кнопки, чекбоксы, селекторы и другие поля). Его задача – помочь пользователю организовать взаимодействие с программой/сайтом. На сегодняшний момент существуют некоторые правила UI дизайна:

- Организованность элементов интерфейса. Это означает, что они должны быть логически структурированы и взаимосвязаны.
- Группировка элементов интерфейса. Подразумевает объединение в группы логически связанных элементов (меню, формы).
- Выравнивание элементов интерфейса. Сложно представить, что плохо выровненный интерфейс может быть для кого-то удобным!
- Единый стиль элементов интерфейса. Стилевое оформление играет не последнюю роль, ведь именно оно сохраняется в памяти пользователя.
- Наличие свободного пространства. Это позволяет разграничивать информационные блоки, сосредотачивая внимание на чем-то одном.

Разработанный по все правилам пользовательский интерфейс значительно повышает эффективность ресурса и дает ему конкурентные преимущества.

### **Сбор функциональных требований (первоначальное проектирование)**

Важность этого этапа трудно переоценить. На нем закладываются основные концепции системы, влияющие абсолютно на все показатели качества её интерфейса. Структурные проблемы практически не могут быть обнаружены и решены на остальных этапах (для их обнаружения нужно слишком много везения, а для исправления – денег). Это значит, что чем больше внимания будет уделено проектированию, тем выше будет общее качество.

Прежде чем начать любое проектирование необходимо выполнить исследование предметной области, которое позволяет проекти-

ровщикам лучше понять цели бизнеса, атрибуты бренда и технические ограничения. Необходимо посмотреть, как организована работа на аналогичных проектах, проанализировать сильные и слабые стороны этих работ, насколько удобно все сделано, насколько грамотно подана информация.

Основной задачей первоначального проектирования является определение необходимой функциональности системы.

Современная наука выдвинула два основных способа определения функциональности, а именно **анализ целей** и **анализ действий пользователей**.

### **Анализ целей пользователей**

Хорошо сформулированная цель должна быть:

- **Понятной.** Избегайте использования узко специализированной терминологии.
- **Ясной.** Избегайте туманных формулировок; подбирайте выражения, которые были бы уместными при определении приоритетов требований.
- **Измеримой.** Используйте конкретные утверждения, которые можно проверить независимо, чтобы определить степень успешности проекта.

Идея, лежащая в основе – *«людям не нужны инструменты сами по себе, нужны лишь результаты их работы»*. Никому не нужен текстовый процессор – нужна возможность с удобством писать тексты.

Очень важно различать задачи и цели. Программисты часто путают их. Программисты обычно занимаются проектированием, ориентированным на задачи. Они начинают проектирование с вопроса: *«Каковы задачи?»*. Компьютерное программирование, если добаться до сути, – это создание подробных пошаговых описаний процедур. Процедура есть рецепт решения задачи. Хорошие программисты имеют «процедурный» взгляд на вещи, взгляд, ориентированный на решение задач.

Такой подход дает возможность сделать работу, но не позволяет даже приблизиться к наилучшему решению, а также совершенно не удовлетворяет пользователя.

**Цель** – это конечное состояние, тогда как **задача** – переходный процесс, необходимый для достижения цели. **Цель** – стабильная сущность. **Задачи** – преходящи.

Вопрос *«каковы цели пользователя?»* позволяет создавать более качественный и уместный дизайн. Для достижения целей пользовате-



ля, конечно, необходимо решать и задачи, однако существуют различные акценты и различные последовательности выполнения задач.

Когда для решения поставленных проблем проектировщики взаимодействия анализируют цели, они обычно находят совсем иные, более подходящие решения.

Рассмотрим цели более подробно. Существуют следующие виды целей: личные, практические, корпоративные, ложные.

**Личные цели:**

- не чувствовать себя глупо;
- не совершать ошибок;
- выполнить адекватный объем работы;
- развлечься (или хотя бы не страдать от скуки).

Личные цели всегда истинны и действительны в определенных рамках для всех людей. Личные цели всегда предшествуют всем другим целям, хотя очень редко становятся предметом обсуждения – как раз потому, что являются личными.

*Любая система, идущая вразрез с личными целями, в конечном итоге обречена на неудачу, независимо от того, насколько качественно позволяет достигать целей иного рода.*

**Корпоративные цели.**

У каждого делового предприятия свои требования к программному обеспечению и уровень этих требований достаточно высок:

- увеличить прибыль;
- увеличить рыночную долю;
- победить конкурентов;
- нанять больше сотрудников;
- предложить новые продукты и услуги;
- выпустить акции компании в свободное обращение.

Цель «увеличить прибыль» является преобладающей для совета директоров или держателей акций. Эти цели необходимые для эффективной работы, но сами по себе не мотивирующие. С точки зрения корпорации это все важные цели, однако, работу выполняет не корпорация, а люди, а для людей важнее цели личные.

Сущность качественного проектирования взаимодействия состоит в том, чтобы позволить пользователям достигать практических целей, не отказываясь от целей личных.

*Программа, которая не позволяет достичь какой-либо корпоративной или личной цели, потерпит неудачу.*

**Практические цели:**

- удовлетворять требованиям клиента;



- сохранять информацию о заказах клиента;
- создавать математические модели бизнеса.

Практическая цель удовлетворения требований клиента соединяет корпоративную цель (более высокие прибыли) с личной целью пользователя (работать продуктивно).

Обычно программисты создают программное обеспечение, которое замечательно помогает достигать практических целей, но совершенно не способно удовлетворить пользователей. Разумеется, чтобы удовлетворить цели бизнеса, вы должны встроить в программу определенные возможности. Однако, если пользователь не может достичь личных целей, то он не способен эффективно достигать целей компании. Непреложный факт: счастливые и довольные сотрудники наиболее эффективны.

С другой стороны, если программа игнорирует практические цели и служит только целям пользователя, это означает, что вы спроектировали компьютерную игру.

#### **Ложные цели:**

- экономия памяти;
- уменьшение потребности в клавиатурном вводе;
- поддержка работы в браузере;
- простота в освоении;
- обеспечение целостности данных;
- ускорение ввода данных;
- увеличение скорости исполнения программы;
- применение супертехнологии или супервозможностей;
- улучшение внешнего вида;
- сохранение единообразия интерфейса на различных платформах.

Повседневные продукты, основанные на программном обеспечении, создаются на основе ложных целей. Многие из этих целей облегчают задачу создания программ, в чем и заключается цель программиста, и потому их приоритет повышается во вред конечному пользователю. Другие ложные цели связаны с задачами, возможностями и инструментами. Все это средства достижения результатов, но еще не результаты, тогда как цели всегда являются результатами.

После того, как истинные цели пользователей установлены (и доказано, что таких пользователей достаточно много, чтобы оправдать создание системы), приходит время выбирать конкретный способ реализации функции, для чего используется второй метод.

## Анализ действий пользователей

Достижение почти всех целей требует от пользователей совершения определенных действий. Разумеется, эти действия могут различаться при разных способах достижения.

Единственным же способом проверить, нужна функция или нет, является наблюдение за пользователями и анализ их действий.

Поскольку на этом этапе мы узнаём, какая именно функциональность нужна для каждого варианта, можно избрать верный путь по правилу *«чем меньше действий требуется от пользователя, тем лучше»*. Не стоит забывать и про другое правило: *чем меньше функций, тем легче их сделать*.

## Создание пользовательских сценариев

Его цель – написать словесное описание взаимодействия пользователя с системой, не конкретизируя, как именно проходит взаимодействие, но уделяя возможно большее внимание всем целям пользователей. Количество сценариев может быть произвольным, главное, что они должны включать все типы задач, стоящих перед системой, и быть сколько-нибудь реалистичными. Сценарии очень удобно различать по именам участвующих в них вымышленных персонажей.

Алан Купер предложил эффективный инструмент для анализа действий пользователя: это точное описание пользователя продукта и его целей. Для этого мы выдумываем несуществующих пользователей и проектируем для них.

Таких несуществующих пользователей называют **персонажами** (*personas*), и они представляют собой необходимую базу качественного проектирования взаимодействия.

Краткая справка: «персонаж» (*франц. personnage, от лат. persona – личность, лицо*) – действующее лицо пьесы (спектакля), сценария (кинофильма), романа и других художественных произведений.

**Персонажи** – не реальные люди, но они представляют реальных людей в процессе проектирования. Будучи воображаемыми, они, тем не менее, определяются достаточно жестко и точно. На практике действительно выдумываются их имена и личные сведения.

Персонажи определяются своими целями. Цели же, разумеется, определяются персонажами.

**Персонаж должен быть конкретным.** Чем более конкретными мы делаем персонажи, тем более эффективными инструментами проектирования они становятся. Для этого мы выбираем ему имя.

**Персонаж должен быть воображаемым.** Описание его должно быть подробным, а не идеальным. То есть важнее определить персонаж насколько возможно подробно и конкретно, чем создать абсолютно правильный персонаж.

**Изобретенные персонажи уникальны для каждого проекта.**

Точно определенный персонаж дает нам:

- определенность относительно уровня владения пользователем компьютером, поэтому мы перестаем терзаться загадкой, для кого проектировать: для дилетанта или специалиста, дают реалистичный взгляд на уровень подготовленности пользователей;

- возможность объяснять наши решения в области проектирования. Это как проектор, высвечивающий для разработчиков, маркетологов, руководителей очевидную правильность наших решений по проектированию.

Программистам свойствен математический подход, и они естественным образом не склонны рассматривать отдельных пользователей, предпочитая обобщение. Ключ к успеху в том, чтобы заставить программистов поверить в существование и реальность созданных персонажей. Примерив на персонаже продукт или задачу, вы сразу можете понять, удастся ли вам его удовлетворить.

Можно выделить шесть типов персонажей, которые назначаются обычно в таком порядке:

- **Ключевой** задает основную цель в проектировании интерфейса, выбирается методом исключения: цели каждого персонажа рассматриваются в сравнении с целями остальных. Если не очевидно, какой из персонажей является ключевым, это может означать одно из двух: или продукту требуется несколько интерфейсов, каждый из которых предназначен для своего ключевого персонажа (так часто бывает в корпоративных и технических продуктах), или же объем его функциональности слишком широк.

- **Второстепенный** в основном оказывается доволен интерфейсом ключевого персонажа, но имеет дополнительные потребности, которые можно включить в продукт, не нарушая его способности служить ключевому персонажу.

- **Дополнительный** – пользовательский персонаж, не являющийся ни ключевыми, ни второстепенным. Их нужды обычно полностью представлены сочетанием нужд ключевого и второстепенных персонажей и удовлетворяются одним из ключевых интерфейсов.

- **Покупатель** – персонаж, отражающий потребности покупателей, а не конечных пользователей. Обычно персонажи покупателей

используются в качестве второстепенных персонажей. Однако в некоторых корпоративных средах кто-то из таких персонажей может оказаться ключевым, если ему предназначается собственный административный интерфейс.

- **Обслуживаемый** не является пользователем продукта, однако его непосредственно затрагивает применение продукта. Обслуживаемые персонажи – это способ отслеживать социальные и физические воздействия второго порядка, оказываемые продуктом. Эти персонажи используются так же, как второстепенные персонажи.

- **Отвергаемый** – используется, чтобы демонстрировать заинтересованным лицам и участникам разработки, что существуют пользователи, для которых продукт не предназначен.

## Подбор персонажей

Каждый проект получает собственный набор персонажей в количестве от трех до двенадцати. Мы проектируем не для каждого из них, но все персонажи полезны для выражения пользовательской аудитории. В каждом наборе персонажей есть хотя бы один ключевой персонаж. Эта личность находится в фокусе процесса проектирования.

Персонажи и цели неразделимы, они – как разные стороны одной медали. Персонаж существует, потому что у него есть цели, а цели существуют, чтобы придавать смысл персонажу.

Следует отметить, что набор характеристик, подробно описывающий пользователя, зависит от предметной области и контекста решаемых им задач. Наиболее общий шаблон профиля содержит в себе следующие разделы:

- социальные характеристики (фотография, имя, возраст, место жительства, род занятий и биография);
- навыки и умения работы с компьютером;
- мотивационно-целевая среда;
- рабочая среда;
- особенности взаимодействия с компьютером (специфические требования пользователей, необходимые информационные технологии и др.).

**Фотография.** Выбирая фотографию, следите за тем, чтобы изображение не казалось постановочным или «глянцевым». Персонаж, снятый в естественной обстановке, создает более яркий и достоверный образ.

**Имя.** С внешним обликом необходимо связать имя. «Нинель» не только лучше звучит, чем «Блондинка, за 30, работает с детьми», но и проще запоминается и связывается с конкретным персонажем. Не

следует использовать имена коллег или заказчиков, хотя такая идея и выглядит соблазнительно.

**Возраст.** Между моделями поведения 21-летней студентки и 34-летней домохозяйки существуют серьезные различия!

**Место жительства.** В Италии, например, в разных областях страны говорят на разных диалектах. В РБ стоимость потребительской корзины жителя Минска, скорее всего, будет отличаться от стоимости потребительской корзины жителя Крупок.

**Род занятий.** Представление о том, чем ваш персонаж зарабатывает себе на жизнь, поможет вам лучше его понять, так как вы сможете опираться на типичные события и ситуации его повседневной жизни. Персонаж, работающий в поликлинике, ежедневно контактирует со многими людьми, тогда как в жизни оператора подъемного крана вряд ли есть избыток общения.

**Биография.** Биография – убедительная история, которая делает персонаж реалистичным. Биография должна быть достоверной, так что наделение персонажа чертами реальных людей – вполне допустимый прием. Сделайте все необходимое, чтобы персонаж выглядел достоверно и как можно более осмысленно для проекта, над которым вы работаете.

Профили пользователей могут по необходимости расширяться за счет добавления других (значимых с точки зрения проектировщика) характеристик пользователей.

При создании персонажей необходимо предоставить достаточно информации, чтобы увлечь людей и заставить их почувствовать человека, описание которого они читают.

**Дополнительный контент.** Базовые подразделы – своего рода «необходимый минимум» для всех создаваемых вами персонажей. В большинстве случаев вам придется добавить к ним те или иные дополнительные элементы. Ценность ваших персонажей можно повысить, если дополнить их описание следующими подразделами:

- **Образование.** Человек с аттестатом средней школы может серьезно отличаться своим покупательским поведением и восприятием бренда от человека со степенью магистра; таким образом, эта информация способна повлиять на то, как будет восприниматься ваш персонаж.

- **Заработная плата.** Эта информация способна привести к значимым открытиям, если вы ориентируетесь на определенные уровни состоятельности.

- **Интернет-активность.** Учитывая, что сейчас многие проекты имеют интернет-составляющую, при подготовке этого элемента следует руководствоваться здравым смыслом.

• **Ключевая точка взаимодействия с заказчиком, брендом или проектом.** Персонаж узнал о них от своих знакомых, по телевизору или радио, прочитал в интернет-обзоре, на форуме или во всплывающем рекламном окне? Используйте статистические данные для прояснения этого момента и включите результат в описание персонажа - это поможет заложить основу для привлечения пользователей к проекту.

• **Техническая подготовка.** На каком компьютере работает ваш персонаж - на PC или на Mac? У него есть собственный компьютер? Использует ли он системы мгновенного обмена сообщениями, Flickr, ведет ли блог? Насколько уверенно чувствует себя при этом? Поможет ли ему очень простое решение, рассчитанное на новичка? Есть ли у него MP3-плеер или другое портативное устройство? Использует ли он DVR, AppleTV или другие устройства для просмотра телепрограмм? Этот список может быть очень длинным. В зависимости от заказчика, бренда или проекта такие мелочи могут сыграть важную роль.

• **Уровень социального комфорта.** Учитывая бурный рост сетевых сообществ и социальных сетей, точное описание степени участия персонажа в сетевом пространстве может многое рассказать о нем. Есть ли у него учетная запись Twitter? Если есть, то сколько у него подписчиков? Насколько персонаж активен? Является ли он лидером? Использует ли MySpace, Facebook, LinkedIn, другие агрегаторы или сетевые сообщества?

## **Разработка сценариев**

На данном этапе уточняется, какими должны быть информация и функциональные возможности интерфейса, чтобы пользователь дошел до целевого действия.

Чтобы создать сценарий, проанализируйте следующие вопросы:

- Кто является главным пользователем в этом сценарии?
- Посещал ли выбранный пользователь этот сайт ранее (работал ли он ранее с данным программным продуктом)?
- Какие срочные потребности привели пользователя на сайт (либо заставили обратиться к данной программе)?

Предположим, что необходимо разработать сценарии для будущей почтовой программы. Судя по всему для этой задачи необходимо три сценария:

А) Елизавета Бронеславовна запускает почтовую программу. Она включает процесс скачивания новой почты. Получив почту, она чита-



ет все сообщения, затем часть их удаляет, а на одно сообщение отвечает. После чего выключает почтовую программу.

Б) Еремей Карпович делает активным окно уже открытой почтовой программы и включает процесс скачивания новой почты. Получив почту, он ее читает. Одно сообщение он пересылает другому адресату, после чего удаляет его, а еще одно печатает. После чего переключается на другую задачу.

С) Пришло новое сообщение, и Эмма Валериевна восприняла соответствующий индикатор. Она делает активным окно почтовой программы и открывает полученное сообщение. Она читает его, после чего перемещает его в другую папку. После чего переключается на другую задачу.

Дело в том, что на таких сценариях очень хорошо заметны ненужные шаги, например, в третьем сценарии гипотетическая Эмма Валериевна после получения индикатора не смогла сразу же открыть новое сообщение, но должна была открыть окно системы, найти нужное сообщение, открыть его и только тогда прочесть. Понятно, что от этих ненужных этапов смело можно избавиться уже на этой, весьма ранней, стадии проектирования.

Польза этих сценариев двояка. Во-первых, они будут полезны для последующего тестирования. Во-вторых, сам факт их написания обычно (если не всегда) приводит к лучшему пониманию устройства проектируемой системы, побуждая сразу же оптимизировать будущее взаимодействие.

## **Проектирование общей структуры**

Итак, информация о будущей системе собрана. Теперь, пользуясь этой информацией, необходимо создать общую структуру системы (*«вид с высоты птичьего полета»*), т.е. выделить отдельные функциональные блоки и определить, как именно эти блоки связываются между собой. Под отдельным функциональным блоком будем понимать функцию/группу функций, связанных по назначению или области применения в случае программы и группу функций/фрагментов информационного наполнения в случае сайта.

Типичная структура сайта (слева) и типичная структура программы представлены на рисунке 9. Если сайты обычно разветвлены, в том смысле, что функции обычно размещаются в отдельных экранах, то программы обычно имеют только один изменяющийся экран, в котором и вызываются почти все функции.

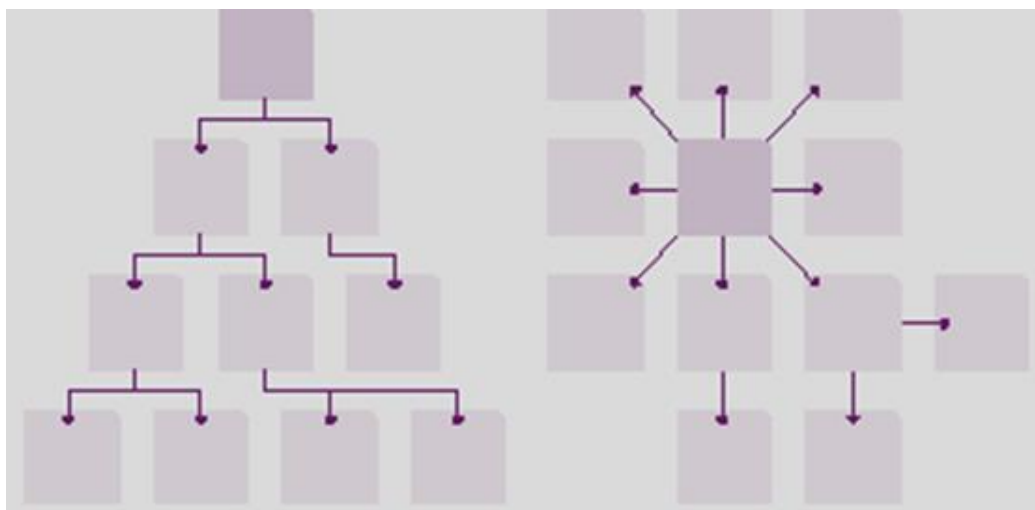


Рисунок 9

Проектирование общей структуры состоит из двух параллельно происходящих процессов: выделения независимых блоков и определения связи между ними. Если проектируется сайт, в завершении необходимо также создать схему навигации.

### Выделение независимых блоков

Для этой работы трудно дать какие-либо конкретные рекомендации, поскольку очень многое зависит от проектируемой системы. Тем не менее, можно с уверенностью рекомендовать избегать помещения в один блок более трех функций, поскольку каждый блок в результирующей системе будет заключен в отдельный экран или группу управляющих элементов. Перегружать же интерфейс опасно. Результатом этой работы должен быть список блоков с необходимыми пояснениями.

В качестве примера рассмотрим гипотетическую программу ввода данных. От пользователя требуется выбрать из списка клиента (или добавить в список нового) и указать, какие именно товары клиент заказал (товары в список тоже можно добавлять).

Несколько клиентов постоянно что-то заказывают, так что заставлять пользователя каждый раз искать в списке такого клиента неправильно. При этом блоки разделяются следующим образом (табл. 2):

Таблица 2

Основной экран	Навигация между функциями системы
Создание нового заказа	

Добавление существующего товара в заказ	А так же простой поиск товара в списке
Сложный поиск товара	
Добавление нового товара в список	
Добавление существующего клиента в заказ	А так же простой поиск клиента в списке
Добавление нового клиента в список	
Выбор постоянного клиента	
Обработка заказа	Печать и его переход в папку «Исполняемые»

### Определение смысловой связи между блоками

Существует три основных вида связи между блоками – логическая связь; связь по представлению пользователей; процессуальная связь.

**Логическая связь** определяет взаимодействие между фрагментами системы с точки зрения разработчика (суперпользователя). Данная связь очень существенно влияет на навигацию в пределах системы (особенно, когда система многооконная). Чтобы не перегружать интерфейс стоит избегать блоков, связанных с большим количеством других – оптимальным числом связей является число три.

**Связь по представлению пользователей.** В информационных системах, когда необходимо гарантировать, что пользователь найдет всю нужную ему информацию, необходимо устанавливать связи между блоками, основываясь не только на точке зрения разработчика, но и на представлениях пользователей.

Например, нужно как-то классифицировать съедобные растения. Помидор, который почти все считают овощем, на самом деле ягода. Не менее тяжело признать ягодой арбуз. Это значит, что классификация, приемлемая для ботаника, не будет работать для всех остальных, причем обратное не менее справедливо.

**Процессуальная связь** описывает пусть не вполне логичное, но естественное для имеющегося процесса взаимодействия: например, логика напрямую не командует людям сначала приготовить обед, а потом съесть его, но обычно получается именно так.

Жестко заданная процессуальная связь позволяет также уменьшить количество ошибок, поскольку от пользователя при ней не требуется спрашивать себя «не забыл ли я чего?».

Замечательным примером жестко заданной процессуальной связи является устройство мастеров (wizards), при котором пользователя заставляют нажимать кнопку «Далее».

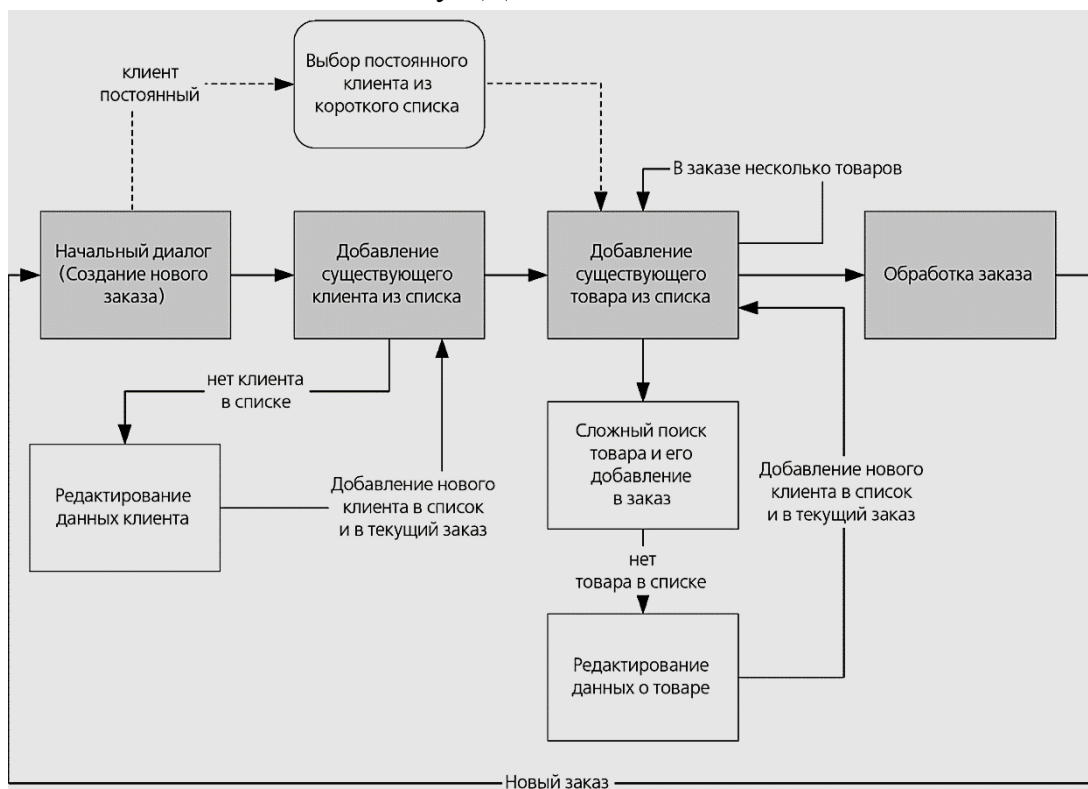


Рисунок 10

Для приведенного выше примера программы ввода данных построена следующая функциональная схема (рис. 10).

Прямоугольник обозначает отдельный экран, прямоугольник со скругленными углами – область экрана, пунктирная линия – альтернативное действие. Обратите внимание, что в этой схеме интерфейс заставляет пользователя выполнять задачу в сугубо определенной последовательности.

Существует любопытная закономерность: чем эстетически привлекательней выглядит схема (без учета цветового кодирования и веселеньких шрифтов), тем она эффективней. Всегда надо стараться сделать схему возможно более стройной и ясной.

Здесь предусмотрено свободное взаимодействие человека и системы (пользователь волен выполнять работу в произвольной последовательности). Рисовать такие схемы очень удобно в MS Visio или подоб-

ной ей системе. Результат, в особенности сложно выглядящий и напечатанный на листе бумаги большого формата, очень хорошо смотрится на стене и животворяще действует на заказчиков и руководство.

### **Создание глоссария**

Еще в процессе проектирования полезно зафиксировать все используемые в системе понятия. Для этого нужно просмотреть все созданные экраны и выписать из них все уникальные понятия (например, текст с кнопок, названия элементов меню и окон, названия режимов и т.д.). После этого к получившемуся списку нужно добавить определения всех концепций системы (например, книга или изображение). Затем этот список нужно улучшить:

- уменьшить длину всех получившихся элементов;
- показать этот список любому потенциальному пользователю системы и спросить его, как он понимает каждый элемент. если текст какого-то элемента воспринимается неправильно, его нужно заменить;
- проверить, что одно и то же понятие не называется в разных местах по-разному;
- проверить текст на совпадение стиля с официальным для выбранной платформы (если вы делаете программу, эталоном является текст из MS Windows);
- убедиться, что на всех командных кнопках стоят глаголы-инфинитивы (задушить, отравить, выкинуть из окна).

После чего список стараться не менять в будущем.

## ЛЕКЦИЯ № 4. ПОСТРОЕНИЕ ПРОТОТИПА. ТЕСТИРОВАНИЕ ПРОТОТИПА

Итак, первый этап пройден. У вас есть полная схема, описывающая всё взаимодействие пользователя с системой. Мы знаем, сколько экранов (страниц) нам нужно и что должно находиться на каждом экране.

Но даже самые блестящие специалисты допускают ошибки. Это правило особенно очевидно в командной работе. «Умные» команды устраняют все ошибки до того, как продукт попадет в руки пользователя, используя методiku, которая называется разработкой прототипов пользовательского интерфейса. В сочетании с тестированием удобства использования продукта (юзабилити-тестирование), прототипы не дают команде сбиться с верного курса.

Разработка прототипа – средство, позволяющее проанализировать идеи, прежде чем потратить на них время и деньги. Все опытные мастера и инженеры создают образцы своих изделий до того, как начинают что-либо строить. Архитектор создаёт модель из бумаги или картона, либо с помощью виртуальных инструментов. Авиаинженеры используют аэродинамическую трубу. Строители мостов разрабатывают модели для исследования нагрузки. Разработчики ПО и веб-дизайнеры создают модели, имитирующие взаимодействие пользователя с их разработками.

Самая веская причина для создания прототипа – экономия времени и ресурсов. Ценность прототипа заключается в том, что он является внешней оболочкой – как декорации на съёмочной площадке в Голливуде, где построены только фасады зданий. По сравнению с реальным продуктом прототипы просты и недороги в разработке. Таким образом, при минимальном вложении средств можно обнаружить ошибки создателей и юзабилити проблемы, и улучшить пользовательский интерфейс до того, как сделаны значительные инвестиции в окончательную разработку и технологии.

**Прототип страниц** – это схематическое представление всех компонентов страницы и их взаимного расположения.

Прототипы используются для достижения одной или нескольких из следующих целей:

- проработать дизайн;
- построить общую коммуникационную платформу;
- увлечь других людей вашими идеями (например, руководство, других проектировщиков и т. д.);
- проверить техническую реализуемость;



- протестировать проектировочные идеи с помощью пользователей (клиентов).

В большинстве случаев прототип после тестирования оказывается неправильным и его приходится переделывать, причем иногда полностью. Поэтому не следует чрезмерно наводить глянец и стремиться сделать его возможно более похожим на результирующую систему. Первый прототип стоит делать максимально примитивным. Только после того, как тестирование подтверждает его правильность, стоит делать более детализированный прототип.

### **Первая версия. Бумажная**

Необходимо нарисовать на бумаге все экраны и диалоговые окна. Нужно убедиться, что все интерфейсные элементы выглядят единообразно и сколько-нибудь похоже на реальные.

Рукописная схема поможет изучить наиболее важную часть приложения – контент. Понимание возможной схемы взаимодействия пользователя с контентом поможет дать более точную оценку числа страниц/экранов, необходимых в программе.

На первом прототипе вполне можно тестировать восприятие системы пользователем и её основную логику.

Полезность прототипирования на бумаге заключается во-первых, в исключительной простоте модификации по результатам тестирования, а во-вторых, в возможности безболезненно отлавливать представления целевой аудитории.

Разумеется, значение слова «версия», весьма условно. В действительности после обнаружения каждой ошибки схема и прототип исправляются, а тестирование продолжается уже на новом прототипе. Так что на этом этапе прототип может пережить множество исправлений и, соответственно, много версий.

Прототип должен создаваться быстро. Его основная функция – показать часть функциональности в простейшем виде без акцентирования на иконках, цветах, шрифтах и тому подобном. Пример бумажного прототипа представлен на рисунке 11.



Рисунок 11

## Вторая версия – раскадровка

Следующий шаг – создание схемы для каждой страницы перехода (раскадровки).

Раскадровки часто используются в фильмах. Видели ли вы дополнительные возможности на DVD-диске фильма «Властелин колец»? Питер Джексон сделал раскадровки всего фильма, чтобы проиллюстрировать его исполнительным продюсерам и разработать сюжет. Вы можете спросить: что общего между кинематографом и интерактивным взаимодействием? Ответ в том, что и то, и другое – дискретные процессы, растянутые во времени.

Раскадровка – это последовательность зарисовок, показывающих, как пользователь продвигается «сквозь» задачу, используя конкретное устройство. Это могут быть эскизы графического пользовательского интерфейса (GUI), скриншоты или наброски сцен пользовательского взаимодействия с программой или устройством.

Скетч (англ. *sketch* – «эскиз, набросок, зарисовка») позволит «оживить» приложение. А также изучить возможные переходы на страницы (со страниц) приложения и понять большее количество деталей и структуры программы.

Раскадровка является промежуточным этапом между электронным и бумажным вариантами прототипов. Создается она при помощи средств электронного офиса. Для этого точно так же рисуется интерфейс, но уже не на бумаге, а в какой-либо программе, например, Microsoft Visio и Microsoft PowerPoint (рис. 12).

The image shows a wireframe of a web application interface for 'TenPro'. It includes a header with a logo, navigation links, and a main content area with several sections:

- Top Left:** Text 'Здесь вы можете найти отличного ИТ-поставщика или поучаствовать в ИТ-тендере самостоятельно.' followed by links: 'Статьи и документы', 'Ответы на вопросы', 'Предложения по улучшению'.
- Top Center:** 'TenPro' logo.
- Top Right:** 'Зарегистрироваться' link, login fields for 'Логин:' and 'Пароль:', a 'запомните меня' checkbox, an 'OK' button, and a 'Не удаётся войти' link.
- Middle Left:** Section 'Я хочу объявить тендер' with a dropdown for 'В чем состоит задача:', a text field for 'Конкурсная документация:', an 'Обзор...' button, and a 'Опубликовать тендер' button.
- Middle Right:** Section 'Я хочу участвовать в тендере' with a search bar 'Искать тендер по ключевым словам:', a 'Найти' button, and a 'Расширенный поиск' link.
- Bottom Left:** Section 'Последние компании-поставщики' listing three companies with their services:
  - ООО «ПроСофтИнфоТехнологии»<sup>100</sup>: Разработка ПО, разработка веб-сайтов, системная интеграция.
  - ИТ-Солюшнс<sup>100</sup>: Разработка веб-решений, веб-маркетинг, раскрутка, поисковая оптимизация.
  - ЗАО «Мосгорпроект»<sup>100</sup>: Поддержка сетей, администрирование, поддержка ИТ.
- Bottom Right:** Section 'Последние тендеры' listing two tenders:
  - ЗАО «Сельхозмашморепродукты»<sup>708</sup>: Разработка корпоративного веб-сайта, 150 000 руб., 1 сентября — 31 декабря 2009.
  - ОАО «Мобильные инфотехнологии»<sup>311</sup>: Разработка и интеграция CRM-системы, 1 000 000 руб., 12—22 сентября 2009.
  - Российское общество охраны<sup>49</sup>: Подключения к сети интернет офиса.

Рисунок 12

Данный вид решения определяется, как пассивная раскадровка.

### Третья версия – интерактивный или кликабельный прототип (действующая модель пользовательского интерфейса)

Дальнейшим развитием пассивной раскадровки является активная раскадровка с применением средств анимации и т.п. При этом каждый экран получает отдельный слайд, а результат нажатия кнопок имитируется переходами между ними. Интерактивная раскадровка представляет собой электронный прототип.

При помощи программных инструментов (вроде Axure RP Pro, Microsoft Expression Blend или плагина к MS Visio Intuitect) вы сможете точно показать, как интерактивные части сайта или приложения будут выглядеть для ваших пользователей.

Существуют следующие подходы к прототипированию: черно-белый прототип на основе схем страниц (wireframe) и цветной прототип, основанный на принятом заказчиком визуальном дизайне системы.

Сделать его можно быстро и дешево, особенно если использовать инструменты быстрого прототипирования.

Кроме этого важно, что получить его можно уже на раннем этапе проектирования, не дожидаясь отрисовки визуального дизайна. А значит, и начать Ю-тестирование – тоже.

С этой версией прототипа можно тестировать значительно более сложное взаимодействие человека с системой, нежели с бумажной. С другой стороны, исправление найденных ошибок значительно более трудоемко.

Однако, гораздо проще общаться с заказчиком, пользователями и разработчиками, имея на руках модель интерфейса, которая выглядит максимально похоже на финальный результат (рис. 13). Правда, делать цветной прототип может быть достаточно дорого и долго.

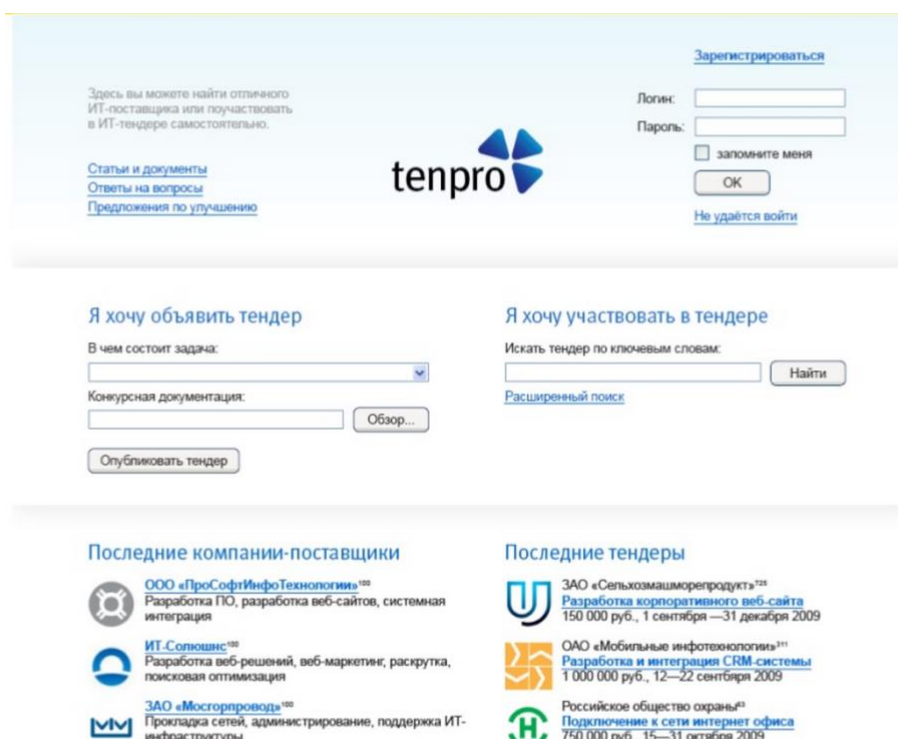


Рисунок 13

## Основные компоновочные блоки макета страницы

К основным блокам можно отнести (рис. 14): навигационные, информационные, сервисные, дизайнерские, рекламные.

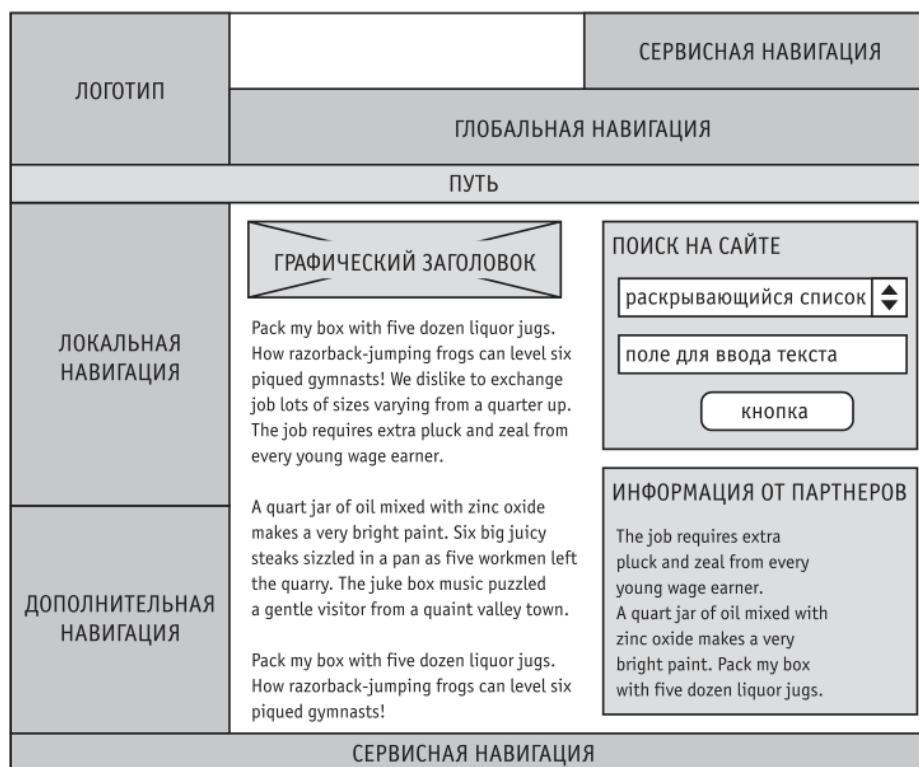


Рисунок 14

**Навигационный блок.** Дизайн навигации кажется простым делом: нужно всего-то расставить на каждой странице ссылки, чтобы пользователь смог ориентироваться на ней. Однако если заглянуть чуть глубже, трудности навигационного дизайна станут очевидными.

Дизайн навигации должен одновременно решать три задачи:

- Предоставлять пользователям способ попасть из одной точки страницы в другую. Поскольку во многих случаях связать каждую страницу со всеми остальными невозможно (а если и возможно, то из общих соображений неразумно), приходится подбирать навигационные элементы так, чтобы они упрощали реальные передвижения пользователя; в числе прочего это подразумевает, что ссылки должны быть рабочими.

- Отражать взаимоотношения между внутренними элементами навигации. Просто предоставить список ссылок недостаточно. Как эти ссылки соотносятся друг с другом? Являются ли одни более важными, чем другие? Какая между ними разница? Эта информация необходима пользователю для понимания того, какой у него есть выбор.

- Отражать связь между содержательной стороной элементов навигации и страницей, которая находится перед глазами пользователя. Какое отношение имеет вся эта куча ссылок к странице, на которую я сейчас смотрю? Эта информация поможет пользователю по-

нять, какой выбор ему следует сделать, чтобы наилучшим образом достичь своей цели или решить стоящую перед ним задачу.

Навигационные блоки:

- «На главную страницу». Данный блок представляет собой гиперссылку, оформленную в виде логотипа либо названия компании. Он чаще всего ведет пользователя на главную страницу. Блок располагается чаще всего в левой верхней части страниц и должен быть повторен на всех страницах сайта, но на небольших персональных сайтах может и вовсе отсутствовать.

- Поиска и быстрого перехода. Несмотря на то что эти функции разные, их можно объединить в один блок из-за их внешней схожести и некоторой схожести по сути. Оба варианта представляют собой поле ввода (редактируемое или нет) с кнопкой выполнения введенного запроса. Наиболее часто блоки поиска и быстрого перехода располагаются в верхней правой части страницы. Они важны для навигации, поэтому «прятать» их от посетителя не рекомендуется, особенно если на страницах много информации.

- Горизонтального меню. Один из самых главных блоков на странице. Под блоком горизонтального меню понимается список гиперссылок, ведущих к основным разделам сайта. Гиперссылки располагаются на одной горизонтальной линии и могут быть оформлены и как обычный текст (меню), и как текст в виде изображения, и как символ (домик, корзина, конверт), и как вкладки. Они могут - быть любого цвета и размера. Горизонтальное меню может быть двухуровневым.

- Вертикального меню. Блок вертикального меню по смыслу соответствует горизонтальному. В силу привычности данный блок располагают по левому краю страницы, но он может находиться и справа, и с обеих сторон одновременно. Этот блок может быть, как статичным, так и с выпадающим меню или раскрывающейся древовидной структурой.

- Вторичной навигации. Визуально представляет собой усеченный вариант горизонтального или вертикального меню. С точки зрения информационного наполнения сайта этот блок не является главным и чаще всего содержит сведения о компании-владельце сайта.

- Навигации по выборке. Этот блок необходим, когда мы работаем с выборкой некоторых объектов (изображения, ссылки, результаты поиска), которую невозможно отобразить целиком. Для перемещения между частями выборки используется специальная навигация. Для удобства пользователя должен быть визуально выделен текущий фрагмент, должны отображаться соседние элементы. Если не хватает



места для отображения всех порций, то следует использовать стрелки или надписи Следующая/Предыдущая.

- Авторизации. Этот блок, наверное самый предсказуемый и понятный. Он располагается там, где пользователь должен идентифицировать себя, чтобы получить доступ к определенной информации или чтобы система могла распознать его и основывать свою работу на ранее введенных данных (например, для очередного заказа в интернет-магазине).

- «Подвал» (текстовые гиперссылки). «Подвал» используется для текстовых гиперссылок на основные разделы сайта. Он должен быстро загружаться и быть доступным в момент, когда фокус внимания находится внизу страницы. Достаточно удобно: закончил работу с этой информацией и готов перейти к следующей. Часто под «подвалом» понимают все, что находится внизу страницы, а не только список основных разделов.

- Навигационной строки. Этот блок – последовательность гиперссылок, определяющая путь посетителя к текущей странице. Навигационная строка показывает, где был пользователь, и позволяет быстро вернуться на один и более шагов назад. Если список элементов становится слишком длинным, то можно отображать только несколько первых и последних ссылок, а промежуточные заменить на многоточие.

**Информационный блок.** Часто обновляемую информацию удобнее всего размещать в виде информационных блоков. Такой способ публикации облегчает задачу добавления и обновления информации за счет структурирования данных, а также за счет использования возможностей импорта и экспорта.

Информационные блоки:

- Содержания. Блок содержания состоит из заголовка и одного или нескольких блоков с основным содержанием страницы.

- Текущей информации. Этот блок используется при необходимости сообщить посетителю какие-то краткие сведения, полезные для его работы. Как правило, такие блоки маленьких размеров и располагаются относительно произвольно, часто внутри других блоков.

- «Раздел». Данный блок достаточно обособленный и содержит информацию, несколько отличающуюся от основного содержания страницы. Это может быть анонс, новость, краткое описание услуг компании, опрос и т. д.. В блоке должно быть название и содержание раздела, а может еще и гиперссылка, позволяющая перейти к полному

содержанию, например; Смотреть новость полностью. Содержимым блока может быть и текст, и набор гиперссылок, и изображение.

- Изображений (галерея). Это набор изображений, совмещенный с блоком навигации по выборке, иногда может быть только одно изображение. Изображения не всегда загружаются достаточно быстро. Однако по непонятной причине на многих сайтах этот блок состоит из одного большого изображения, вместо нескольких маленьких, которые можно просмотреть в большем размере дополнительным шагом.

Лучше делать картинки двух размеров. Т. е. делать два вида блока – для множества картинок и для одной. Если пользователь не хочет внимательно рассматривать каждую картинку из шести представленных, то он дождется загрузки шести маленьких изображений, а не вынужден будет ждать загрузки страницы с полномасштабным изображением, что в результате вызовет его раздражение.

Информационный дизайн играет важную роль и в тех задачах, где интерфейс должен не только получать какие-то сведения от пользователя, но и передавать ему информацию. Классическая задача информационного дизайна при создании успешных интерфейсов – *сообщения об ошибках*; еще одна – *предоставление инструкций пользователю* (задача непростая уже хотя бы потому, что труднее всего заставить их прочитать эти инструкции). Каждый раз, когда система должна облегчить пользователю работу с интерфейсом путем предоставления информации (например, когда пользователь только начал работать с сайтом или совершил ошибку), это задача информационного дизайна

#### **Сервисные блоки:**

- «Выбор языка». Этот блок необходим, если ваш сайт поддерживает несколько языков, Часто его располагают рядом со вторичной навигацией, а наименование языка пишут на нем же самом.

- «Пустой блок». Этот блок представляет собой пустое место между другими и может служить для отделения одного блока от другого, для рекламного баннера. Чаще всего он появляется сам по себе в процессе верстки (например, пустое место под горизонтальным меню, когда меню уже закончилось, а основное содержание страницы располагается ниже).

- «Версия для печати». Наверное, наименьший по размерам блок, он инициирует вызов текущей страницы и оптимизирует ее для отправки на принтер средствами браузера. Как правило, он располагается вверху или внизу основного содержимого страницы и совмещен с пиктограммой принтера.

**Дизайнерский блок.** Изображение, созданное дизайнером для украшения сайта и не являющееся одним из основных элементов содержания, может быть использовано и в качестве фона для иных блоков, например, для блока «Название и слоган» или блока текущей информации.

**Рекламный блок.** Несмотря на то, что такой блок напоминает «Раздел», он может выглядеть как угодно: тут все зависит от фантазии разработчика. Это может быть и мерцающий баннер на половину экрана, и маленькая гиперссылка-«завлекалка», и имитация кнопки, например, «Рейтингуются SpyLog» или «Участник Rambler Top 100». Реклама – она и есть реклама – всегда стремится к разнообразию.

К рекламным блокам можно отнести и:

- «Название и слоган». Этот блок содержит название компании или самого сайта и, возможно, слоган вроде «Уважаем классику, ценим новое». Этот блок оформляется чаще всего крупным шрифтом, иногда на фоновом рисунке или вместе с логотипом компании. Назначение его, думаю, понятно – это громкое заявление компании о себе.

- «Копирайт». Блок описания авторских прав компании-владельца сайта, иногда со ссылкой на его создателя, например «Дизайн SuperWeb-Studio».

## **Инструменты для создания прототипов**

Существует много инструментов для создания прототипов и дизайна интерфейса это –Axure RP Pro, Microsoft Expression Blend или плагина к MS Visio Intuitect. Рассмотрим некоторые из них.

Нужно определиться в самом начале, что в проекте является эталоном интерфейса – схемы страниц (wireframes), макеты дизайна или интерактивный прототип.

Для простых задач можно использовать любое средство, чтобы рисовать:

- **Microsoft PowerPoint** (<http://www.powermockup.com/>). В PowerPoint существует плагин под названием PowerMockup. Доступ к PowerMockup можно получить через панель инструментов, устанавливающуюся вместе с библиотекой. Все базовые элементы присутствуют. Есть даже шаблоны, изображающие корпуса смартфона и планшета. Хотя эта библиотека и уступает по числу полезных шаблонов лучшим программам, в ней есть все, что нужно для работы на скорую руку. Разумеется, можно изменять шаблоны в размерах и редактировать текст, который на них присутствует. А вот изменять состояние элементов управления, например, нажать кнопку или выбрать

один из пунктов выпадающего меню, как это позволяют делать полноценные программы, к сожалению, не получится. Доступ к PowerMockup можно получить через панель инструментов, устанавливающуюся вместе с библиотекой.

• **Microsoft Visio 2013.** Visio 2013 имеет родной набор шаблонов для создания схем интерфейсов. Для того чтобы им воспользоваться, нужно при создании нового файла выбрать из категории «Программное обеспечение» пункт «Проволочная диаграмма». Штатный набор шаблонов Visio более полон и включает в себя почти все, что есть в специализированных программах, кроме макетов для мобильных приложений. В отличие от плагина для PowerPoint, этот набор шаблонов доступен в качестве родного для Visio файла с расширением vss и не пытается устанавливаться в системе как отдельная программа. Однако, в Visio отсутствует возможность изменять состояния элементов управления. Например, отсутствует такой элемент, как пустая (не выбранная) радиокнопка. А чтобы создать раскрывающийся список, придется использовать отдельные шаблоны для кнопки со стрелкой и каждого из элементов списка. Все это приводит к тому, что процесс создания макета в Visio занимает заметно больше времени, чем в специализированной программе.

Однако специальное, всегда лучше универсального. Поэтому рассмотрим специализированные пакеты.

**Axuro RP Pro** – ориентирована на создание веб-сайтов. Генерирует кликабельный HTML- документ и документацию в формате MS Word.

**Caretta GUI Design Studio** – специализированное средство, позволяющее создавать интерфейсы в разных визуальных стилях, аннотации к ним, раскладки и т. п. Можно экспортировать прототип.

**Balsaming Mockups** – подходит для быстрого создания макетов интерфейсов. Прототипы выглядят рисованными.

**Adobe InDesign** – изначально рассчитан для верстки полиграфических материалов, тем не менее подходит и для отрисовки прототипов. На выходе: кликабельный PDF.

**Adobe Fireworks** – специально предназначен для прототипирования интерфейсов. На выходе: кликабельный PDF или HTML.

**Adobe Dreamweaver** – предназначен для HTML-верстки. На выходе: кликабельный HTML. Интегрируется с Visual Studio. Прототип можно преобразовать в конечный продукт. Использует технологии Silverlight или WPF.

**Microsoft Expression Blend.** Интегрируется с Visual Studio. Прототип можно преобразовать в конечный продукт. Использует технологии Silverlight или WPF.

**Microsoft Silverlight** – это программная платформа, включающая в себя модуль для браузера, который позволяет запускать приложения, содержащие анимацию, векторную графику и аудио-видео ролики, что характерно для RIA (Rich Internet application).

**WPF** или полное название – Windows Presentation Foundation. Это система, предназначенная для построения клиентских приложений операционной системы Windows. Она содержит визуально привлекательные возможности для пользователя, среди которых графическая подсистема .Net Framework

Сейчас лучшей технологией прототипирования является сначала рисование прототипа на бумаге, а затем — финализация прототипа в Adobe InDesign – он легче в изучении, чем специализированные средства разработки, к тому же, хоть и уступая им в скорости создания первой версии, лидирует в скорости модификаций. Наконец, InDesign, в отличие от средств разработки вроде Adobe Dreamweaver или Microsoft Visual Studio, универсален – в нем можно запрототипировать любой графический интерфейс, будь то интерфейс программы или сайта.

Для редактирования XAML лучше всего применять специальный пакет, Microsoft Expression Blend или просто Blend. Редактирование внутри него происходит в наглядной, визуальной форме, то есть проектировщик может создавать элементы, перемещать их по экрану и описывать поведение.

### **Тестирование и модификация прототипа**

В некоторых случаях выгодно не заморачиваться с Ю-тестированием прототипа, а быстро запустить бета-версию системы. И уже получая обратную связь, быстро вносить корректировки. Но такой подход больше подходит для проектов с гарантированной и лояльной аудиторией, чем для тиражных продуктов с сильными конкурентами.

Проверка качества интерфейса обычно неproblemатична. Всё, что для этого нужно, это несколько пользователей средней квалификации, никогда не видевшие тестируемой системы, плюс прототип.

В литературе часто встречается мнение, что тестированием можно решить чуть ли не все проблемы интерфейса. Утверждение это сомнительно. Тестированием можно определить слабые места интерфейса, но почти невозможно обнаружить сильные, поскольку они пользова-

телями просто не замечаются, и совсем уж невозможно определить новые способы улучшения. Происходит это из-за того, что субъекты тестирования: не обладают всей необходимой информацией о системе, ничего не знают о проектировании интерфейсов, их мотивация существенно отличается от необходимой – вместо того, чтобы стремиться сделать хороший интерфейс, они стремятся оставить в этом интерфейсе свой след.

Тем не менее, нужно принимать во внимание потребности пользователей во избежание следующих ситуаций:

- Дизайнер интерфейса знает о предметной области меньше, нежели будущие пользователи. В таких случаях система оказывается неспособна решать задачи, о которых дизайнер ничего не знал;

- Уровень «компьютерной грамотности» дизайнера оказывается выше уровня аудитории. В этом случае дизайнер выбирает решения, которые пользователи не могут понять.

Замечено, что опытные пользователи (к которым относятся дизайнеры) создают значительно менее работоспособные иерархии меню, нежели пользователи начинающие.

Если переоценка способностей реальных пользователей и незнание предметной области совпадают, результат бывает ещё хуже.

**Постановка задачи.** Успех тестирования зависит от правильности и корректности постановки задачи тестирования. Тестирование может быть направлено на подтверждение:

- Производительности действий при использовании продукта. Оценивается по длительности выполнения задач (тестовых заданий) пользователем. Эффективный продукт позволяет увеличить число пользователей успешно выполняющих задание в течение ограниченного времени;

- Полезности продукта. Продукт является полезным, если позволяет снизить количество человеческих ошибок. Полезный продукт позволяет увеличить число пользователей, способных успешно выполнить задание;

- Простоты обучения. Оценивается по времени тренинга, необходимого для достижения пользователем определенного уровня владения продуктом;

- Субъективной оценки пользователей. Пользователи оценивают свое отношение к продукту по десятибалльной шкале. Продукт можно считать успешным, если определенная часть пользователей оценила его на 8 и выше баллов.



Тестовые задания, которые в ходе проведения представляют собой задачи для пользователей, формируют исходя из указанных задач тестирования. Основой формулирования тестовых заданий являются пользовательские сценарии.

**Тестирование.** Тестирование проводится на представителях пользовательской аудитории ранее не знакомых с разрабатываемым продуктом. Уровень опытности тестируемых пользователей должен соответствовать уровню, определенному в профилях конечных пользователей. Считается, что тестирование на одном пользователе позволяет выявить примерно 60% ошибок. Поэтому число тестируемых пользователей, необходимых для проведения одного сеанса зависит от сложности и объема проектируемого продукта. Для «средних» приложений достаточно 4-8 человек. В ходе тестирования категорически запрещено прерывать или смущать пользователя; нельзя внушать тестируемому, что тестируют его; желательно присутствие разработчиков приложения (программистов), но их роль в тестировании исключительно пассивная.

В качестве методов проведения тестирования могут быть использованы наиболее простые:

- **Наблюдение за пользователем.** Пользователю предъявляется тестовое задание, он его выполняет. Действия пользователя фиксируются. Этот метод эффективен при определении неоднозначности элементов интерфейса: любая неоднозначность, как правило, влечет за собой ошибку пользователя. Поскольку действия пользователя фиксируются, обнаружить ошибки при анализе тестов довольно легко. Кроме того, этот метод подходит для оценки производительности действий пользователя. Для этого необходимо при фиксировании действий замерять время, потребовавшееся пользователю на его выполнение.

- **Комментарии пользователя.** Как и при использовании предыдущего метода тестирования, пользователи выполняют тестовые задания. Действия пользователя также фиксируются, кроме того, фиксируются комментарии им своих действий. В дальнейшем комментарии позволяют выявить недостатки реализации конкретных элементов интерфейса - неудачное расположение элементов управления, плохая навигация и т.д.

Этот метод можно использовать для оценки полезности продукта, простоты обучения работы с ним, степени субъективного удовлетворения. Следует отметить, что метод является «нестабильным»: результаты его использования зависят от личных качеств тестируемого

пользователя – его разговорчивости, умения последовательно и внятно излагать свои мысли.

• **Качество восприятия.** Пользователю предъявляется тестовое задание, через некоторое время после его выполнения, пользователь должен воспроизвести экранные формы (бумажный вариант), с которыми он работал. Результат воспроизведения сравнивают с оригиналом. Идея теста заключается в следующем. Из-за ограничения на объем кратковременной памяти, количество элементов экранных форм, которые запоминает тестируемый, не может быть выше порога запоминания. Пользователь запоминает только то, что считает наиболее актуальным в процессе работы. Следовательно, при повторном выполнении задания пользователю, знающему расположение необходимых для этого элементов интерфейса, будет проще. Таким образом, этот метод позволяет оценить простоту обучения работе с продуктом, а, кроме того, степень субъективной удовлетворенности пользователей.

**Модификация.** Тестирование само по себе имеет существенный недостаток: если тестирование проблем не выявило, получается, что оно было проведено зря. Если выявило, придется проблемы решать, что тоже существенная работа.

Следует отметить, что выявление в ходе тестирования различных ошибок и несоответствий неизбежно. Это является одной из причин того, что тестирование нельзя переносить на окончание проекта, когда вносить модификации нет возможности из-за истечения сроков работ.

Разработка пользовательского интерфейса приложения представляется собой итеративный процесс. Каждая итерация связана с отдельным этапом проектирования, созданием прототипа по его результатам, тестированием прототипа и его модификацией. Разработчик должен прилагать особые усилия, чтобы уменьшить число итераций. Тестерам дается задание, они его выполняют, после чего результаты анализируются.

Именно поэтому тестирование бессознательно переносят на самое окончание проекта, когда что-либо исправлять уже поздно. В результате тестирование показывает, что проект сделан плохо, что никому не нравится, включая его создателя, после чего результаты проверки прячутся в дальний ящик. В то время как сама по себе идея тестирования совсем иная. В самом начале работы, когда только создан прототип будущей системы, он тестируется, после чего найденные ошибки исправляются. А затем прототип тестируется опять. При этом опытность дизайнера проявляется исключительно в уменьшении количества итераций. Соответственно, тестирование должно идти параллельно со всеми остальными операциями.

## ЛЕКЦИЯ № 5. КРИТЕРИИ КАЧЕСТВА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Существует четыре основных критерия качества любого интерфейса, а именно:

- скорость работы пользователей,
- количество человеческих ошибок,
- скорость обучения,
- субъективное удовлетворение пользователей (подразумевается, что соответствие интерфейса задачам пользователя является неотъемлемым свойством интерфейса).

### **Скорость работы пользователей**

Скорость выполнения работы является важным критерием эффективности интерфейса. Любая попытка как-то увеличить производительность труда всегда встречается положительно.

Длительность выполнения работы пользователем состоит из следующих составных частей:

- длительности восприятия исходной информации;
- длительности интеллектуальной работы (пользователь думает, что он должен сделать);
- длительности физических действий пользователя;
- длительности реакции системы.

**Длительность восприятия исходной информации** в особых комментариях не нуждается, Пользователь должен представить себе, какая информация о выполняемой задаче у него существует, и в каком состоянии находятся средства, с помощью которых он будет решать данную задачу. Основное время здесь пойдет на считывание показаний системы.

**Длительность интеллектуальной работы** – оценивается взаимодействие пользователя с системой (не только компьютерной) и состоит из семи шагов:

1. Формирование цели действий.
2. Определение общей направленности действий.
3. Определение конкретных действий.
4. Выполнение действий.
5. Восприятие нового состояния системы.
6. Интерпретация состояния системы
7. Оценка результата.

Из этого списка становится видно, что процесс размышления занимает почти все время, в течение которого пользователь работает с компьютером, во всяком случае, шесть из семи этапов полностью заняты умственной деятельностью.

К сожалению, существенно повысить скорость собственно мышления пользователей невозможно. Тем не менее, уменьшить влияние факторов, усложняющих и, соответственно, замедляющих процесс мышления, вполне возможно.

Факторы, позволяющие ускорить процесс мышления:

- **Непосредственное манипулирование.** Смысл этого метода очень прост. Пользователь не отдает команды системе, а манипулирует объектами. Первым популярным применением этого метода была корзина для удаления файлов в компьютерах *Macintosh* (начиная с *Windows 95*, такая корзина стала стандартом и в *Windows* мире, хотя присутствовала она и раньше). Если перетащить в неё пиктограмму файла, этот файл будет фактически стерт.

На самом деле процесс стирания файла, состоит из многих малых, уже не делимых, действий (жестов). При этом для ускорения мыслительной работы пользователя необходимо не только сокращать количество этих жестов, но и делать эти жесты более простыми.

- **Применение в интерфейсе эффективных методов при потере фокуса внимания.** При работе с системой, пользователи постоянно отвлекаются. Таким образом, необходимо максимально облегчать возвращение пользователей к работе и проектировать интерфейс так, чтобы пользователи возможно меньше о нем думали.

Итак, для продолжения работы пользователь должен знать:

- На каком шаге он остановился;
- Какие команды и параметры он уже дал системе;
- Что именно он должен сделать на текущем шаге;
- Куда было обращено его внимание на момент отвлечения.

Предоставлять пользователю всю эту информацию лучше всего визуально. Разберем это на примере.

Чтобы показать пользователю, *на каком шаге он остановился*, традиционно используют конструкцию «Страница N из N» (рис. 15).

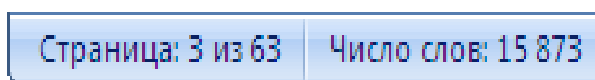


Рисунок 15

К сожалению, эта конструкция работает не слишком эффективно, поскольку не визуальна. Однако существуют и визуальные способы. Например, когда читатель держит в руках книгу, он может понять, в какой её части он находится, по толщине левой и правой части разворота. Можно воспользоваться этой метафорой и на экране: варьировать толщину левых и правых полей окна.

*Показ пользователю ранее отданных им команд.* Размеры экрана ограничены, так что единственным выходом из этого положения является максимальное облегчение перехода к предыдущим экранам, да и то это работает только с экранными формами.

Напротив, показывать пользователю, *что именно он должен сделать на текущем шаге процедуры*, обычно удается легче. С другой стороны, это очень сильно зависит от сущности задачи, так что тут трудно порекомендовать что-либо конкретное.

И, наконец, четвертый пункт: *показ пользователю, куда было обращено его внимание на момент отвлечения*. Обычно фокус внимания совпадает с фокусом ввода. Соответственно, нужно делать фокус ввода максимально более заметным, например цветовым кодированием активного элемента. Если количество элементов на экране невелико, пользователь быстро находит активный элемент. Таким образом, просто снизив насыщенность экрана элементами, можно значительно облегчить пользователю возвращение к работе.

**Длительность физических действий** пользователя зависит от степени автоматизации работы и степени необходимой точности работы.

Об автоматизации что-либо конкретное сказать сложно. Понятно, что чем больше работы делает компьютер, тем лучше.

С точностью все гораздо проще. Любое физическое действие, совершаемое с помощью мускулатуры, может быть *или* точным *или* быстрым. Вместе точность и быстрота встречаются исключительно редко.

Пользователь, как правило, управляет компьютером двумя способами, а именно мышью и клавиатурой. Клавиатура не требует особой точности движений – неважно, быстро нажали клавишу или медленно, равно как сильно или слабо. Мышь, напротив, инерционна. Именно поэтому оптимизация использования мыши в системе может существенно повысить общую скорость работы.

Мышь не предназначена для очень точных, в 1 или 2 пикселя, манипуляций (попробуйте мышью нарисовать ровный круг). Обычно в графических программах всегда есть возможность перемещать объекты клавишами со стрелками. Именно поэтому любой маленький ин-

терфейсный элемент будет всегда вызывать проблемы у пользователей.

Еще в 1954 году Пол Фиттс (Paul Fitts) сформулировал правило, ставшее известным как Закон Фиттса:

***Время достижения цели прямо пропорционально дистанции до цели и обратно пропорционально размеру цели***

$$T_{\text{дост. цели}} = a + b \log_2 \left( \frac{D}{S} + 1 \right) \dots$$

Где  $a$  и  $b$  – устанавливаются опытным путем по параметрам производительности человека. Для практического использования можно принять:  $a = 50$ ,  $b = 150$ ,  $D$  – дистанция от курсора до цели,  $S$  – размер цели по направлению движения курсора.

Популярно говоря, ***лучший способ повысить доступность кнопки заключается в том, чтобы делать её большой и располагать ближе к курсору.***

У этого правила есть два не сразу заметных следствия. Чтобы «бесконечно» ускорить нажатие кнопки, её, во-первых, можно сделать бесконечного размера и, во-вторых, дистанцию до неё можно сделать нулевой.

***Кнопка бесконечного размера.*** При подведении курсора к краю экрана он останавливается, даже если движение мыши продолжается. Это значит, что кнопка, расположенная впрыток к верхнему или нижнему краю экрана, имеет бесконечную высоту (равно как кнопка у левого или правого края имеет бесконечную ширину). Таким образом, скорость достижения такой кнопки зависит только от расстояния до неё и точности выбора начального направления движения. Понятно, что кнопка, расположенная в углу экрана, имеет бесконечные размеры (не важно даже, с какой точностью перемещали мышь). Для достижения такой кнопки от пользователя требуется всего лишь дёрнуть мышь в нужном направлении, не заботясь о её скорости и не делая попыток остановить её в нужном месте. Это делает такие кнопки наиболее доступными для пользователя, жалко даже, что у экрана всего четыре угла.

Именно поэтому, например, меню MacOS многократно эффективней меню Windows: если в MacOS меню всегда расположено впрыток к верхнему краю экрана, то в Windows меню отделено от края экрана полосой заголовка окна программы (Title Bar).

***Нулевая дистанция до кнопки.*** Рассмотрим контекстное меню, вызываемое по нажатию правой кнопки мыши. Оно всегда открывает-



ся под курсором, соответственно расстояние до любого его элемента всегда минимально. Именно поэтому контекстное меню является чуть ли не самым быстрым и эффективным элементом.

**Длительность реакции системы.** Часто пользователи надолго прерывают свою работу. Попросту говоря, система делает что-либо длительное. Например, печать документа в сто страниц даже на быстрых принтерах занимает существенное время, соответственно, большинство людей, отправив такой документ в печать, начинают бездельничать. Проблема в том, что сразу после того, как человек отвлекается, системе зачастую, во что бы то ни стало, начинает требоваться что-либо от человека. Например, появляется диалоговое окно с вопросом «*Вы уверены?*».

Если процесс предположительно будет длительным, система должна убедиться, что она получила всю информацию от пользователя до начала этого процесса.

Есть другое решение этой проблемы: система может считать, что если пользователь не ответил на вопрос, скажем, в течение пяти минут, то его ответ положительный.

Таким образом, тот же самый сценарий решается по другому: пользователь отправляет документ на печать и уходит, система спрашивает «*Вы уверены?*» и ждет пять минут, после истечения этого времени она начинает печать. Этот метод вполне работоспособен, так что им стоит пользоваться всегда, когда невозможен первый метод (разумеется, за исключением случаев, когда ответ «*Да*» может иметь катастрофические последствия).

***Убирайте с экрана все диалоги с вопросами, на которые в течение пяти минут не был дан ответ.***

### **Количество человеческих ошибок**

Важным критерием эффективности интерфейса является количество человеческих ошибок.

Человек при работе с компьютером постоянно совершает ошибки.

Часто минимальная ошибка приводит к совершенно катастрофическим последствиям, например, за одну секунду оператор в банке может сделать кого-то богаче, а банк, в свою очередь, беднее (впрочем, обычно беднее становятся все).

**Типы ошибок.** Классификаций человеческих ошибок существует великое множество. Наибольшее количество человеческих ошибок раскладывается на четыре типа:

• **Ошибки, вызванные недостаточным знанием предметной области.** Теоретически, эти ошибки методологических проблем не вызывают, сравнительно легко исправляясь обучением пользователей. Практически же, роль этих ошибок чрезвычайно велика – никого не удивляет, когда оператора радарной установки перед началом работы оператором долго учат работать, и в то же время все ожидают должного уровня подготовки от пользователей ПО, которых никто никогда ничему целенаправленно не обучал. Еще хуже ситуация с сайтами, у которых даже справочной системы почти никогда не бывает.

• **Опечатки** происходят в двух случаях: когда не все внимание уделяется выполнению текущего действия (этот тип ошибок характерен, прежде всего, для опытных пользователей, не проверяющих каждый свой шаг) и когда в мысленный план выполняемого действия вклинивается фрагмент плана из другого действия (происходит преимущественно в случаях, когда пользователь имеет обдуманное текущее действие и уже обдумывает следующее действие).

• **Ошибки, вызванные не считыванием показаний системы,** которые одинаково охотно производят как опытные, так и неопытные пользователи. Первые не считывают показаний системы потому, что у них уже сложилось мнение о текущем состоянии, и они считают излишним его проверять, вторые – потому что они либо забывают считывать показания, либо не знают, что это нужно делать и как.

• **Моторные ошибки,** количество, которых фактически пренебрежимо мало, но к сожалению, не так мало, чтобы вовсе их не учитывать. Сущностью этих ошибок являются ситуации, когда пользователь знает, что он должен сделать, знает, как этого добиться, но не может выполнить действие нормально из-за того, что физические действия, которые нужно выполнить, выполнить трудно. Так, никто не может с первого раза (и со второго тоже) нажать на экранную кнопку размером 1 на 1 пиксель. При увеличении размеров кнопки вероятность ошибки снижается, но почти никогда не достигает нуля. Соответственно, единственным средством избежать этих ошибок является снижение требований к точности движений пользователя.

Для минимизации количества ошибок нужно:

- плавно обучать пользователей в процессе работы;
- повышать разборчивость и заметность индикаторов состояния;
- снижать чувствительность системы к ошибкам.

Для этого есть три основных способа, а именно:

- блокировка потенциально опасных действий пользователя до получения подтверждения правильности действия

- Проверка системой всех действий пользователя перед их принятием.
- Самостоятельный выбор системой необходимых команд или параметров, при этом от пользователя требуется только проверка.

Самым эффективным является третий способ. К сожалению, этот способ наиболее труден в реализации. Разберем эти три способа подробнее.

**Блокировка потенциально опасных действий до получения подтверждения.** Команда удаления файла в любой операционной системе снабжена требованием подтвердить удаление (рис. 16). Эта блокировка приносит пользу только начинающим пользователям, которые проверяют каждый свой шаг.

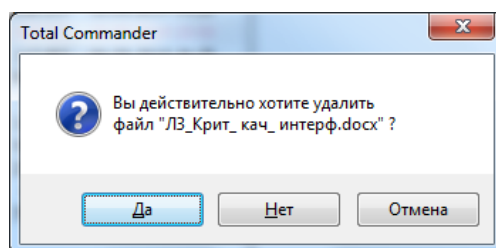


Рисунок 16

Для опытных пользователей это диалоговое окно с требованием подтверждения не работает. Во-первых, оно не защищает нужные файлы. Во-вторых, оно без пользы отвлекает пользователя и тратит его время.

Надо требовать подтверждения не после команды пользователя, а до неё. Например, сначала – **разблокировать**, после чего – **удалить**. Поскольку эти два действия напрямую не связаны друг с другом – если в одном из них была допущена ошибка, файл удалить не удастся.

Гораздо чаще приходится защищать не отдельные объекты (файлы, окна и т.п.), но отдельные фрагменты данных (например, текст и числа в полях ввода). Понятного и удобного элемента управления для этой цели нет. Единственным выходом служит скрывание потенциально опасных данных от пользователя до тех пор, пока он сам не скомандует системе их показать. Выход же этот отнюдь не идеальный, поскольку некоторым пользователям никогда не удастся понять, что, помимо видимых значений, есть еще и невидимые данные.

***Не делайте опасные для пользователя кнопки кнопками по умолчанию.***

**Проверка действий пользователя перед их принятием.** Этот метод гораздо лучше блокировки. Наиболее популярны два универсальных и работающих способа проверки.

Во-первых, это меню. В случаях, когда пользователь выбирает команду из списка, система может без труда делать так, чтобы в этот список попадали только корректные команды (это вообще достоинство любого меню).

Во-вторых, если действие запускается непосредственным манипулированием объектами, можно индцировать возможные действия изменением поведения этих объектов.

Например, если бы форматирование диска запускалось не нажатием кнопки, а перенесением пиктограммы диска в область форматирования, можно было бы показывать пользователю, как с выбранного диска исчезают все файлы и папки. При этом не только снизилась бы вероятность ошибочного форматирования диска, поскольку перенести объект в другую область труднее, чем просто нажать на кнопку, но при этом исчезла бы необходимость предупреждать пользователя о грядущей потере данных.

Проверкой всех действий пользователя перед их принятием можно также успешно защищать вводимые пользователем данные, в особенности данные численные. Дело в том, что большинство численных данных имеют некий диапазон возможных значений, так что даже в ситуациях, когда невозможно проверить корректность данных, можно, по крайней мере, убедиться, что они попадают в нужный диапазон. В большинстве ОС есть специальный элемент управления, именуемый «крутилкой». Фактически это обычное поле ввода, снабженное двумя кнопками для модификации его содержимого (в сторону уменьшения и увеличения). Интересен он тем, что пользователь может не пользоваться клавиатурой для ввода нужного значения, взамен клавиатуры установив нужное значение мышью. Этот элемент имеет то существенное достоинство, что при использовании мыши значение в этом элементе всегда находится в нужном диапазоне и обладает нужным форматом.

***Всегда показывайте границы диапазона во всплывающей подсказке.***

Но что делать, если пользователь ввёл некорректное число с клавиатуры? Ответ прост. Для этого надо индцировать возможную ошибку изменением начертания шрифта на полужирное в обычных программах (иное проблематично), а в случае сайта – заменой цвета фона этого элемента на розовый (благо это нетрудно сделать через таблицу стилей).

В тех же случаях, когда количество возможных значений невелико, лучше использовать другой элемент управления – ползунок. Мало того, что он позволяет устанавливать только определенные значения (с этим справился бы и выпадающий список или комплект переключателей), но он позволяет пользователю видеть взаимосвязь возможных значений и при этом использование этого элемента понятно даже новичку.

**Самостоятельный выбор команд.** Самый эффективный способ. Чем меньше действий требуется совершить пользователю, тем меньше вероятность ошибки.

Проиллюстрировать сферу применения данного метода удобно на примере печати. В MS Word 2003 существует две команды меню «Файл», а именно «Печать» и «Параметры печати», которые вызывают одноименные диалоговые окна (рис. 17).

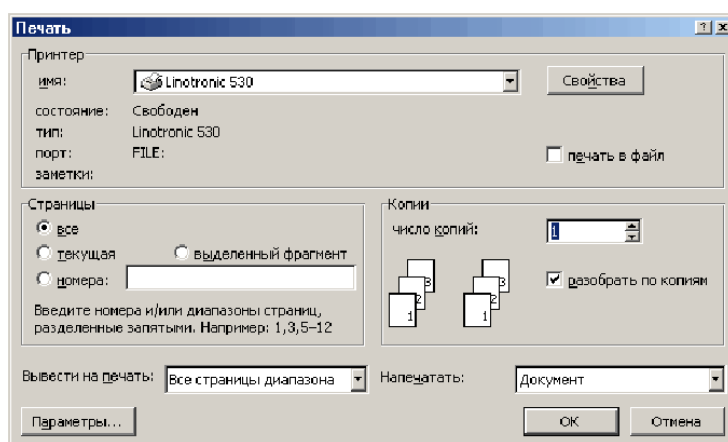


Рисунок 17

Обилие элементов управления замедляет восприятие этих окон и увеличивает вероятность ошибки. Чем меньше элементов управления, тем меньше вероятность ошибки. Главной причиной появления этих диалоговых окон является печать нескольких копий.

Стандартное диалоговое окно печати содержит также область выбора принтера из числа установленных в системе. Большинство же пользователей имеет только один принтер. В каждой включенной в комплект MS Office программе на панели инструментов есть кнопка, нажатие на которую вызывает печать одного экземпляра с текущими настройками. Это, впрочем, тоже нехорошо.

Во-первых, кнопка называется «Печать». Это название конфликтует с такой же командой в меню (называть кнопку «Печать одного экземпляра с текущими настройками» неприлично).

Во-вторых, получается, что в панели инструментов есть команда, которую нельзя вызвать никаким иным способом.

А теперь представьте себе другой вариант. В меню MS Word 2007 есть то же две команды – «Печать» и «Быстрая печать».

Выбор первой вызывает диалог со всеми доступными параметрами печати, т.е. в системе с одним принтером никогда не появится ненужная группа элементов.

Выбор второй команды вызывает немедленную печать документа с текущими настройками. Если же пользователь захочет напечатать несколько копий, включается другой механизм. Программа запоминает его действие и при следующем выборе команды «Печать» выводит диалоговое окно с полем ввода, в котором уже стоит число копий (которое было запомнено в предыдущий раз). Причём такой метод применяется абсолютно ко всем возможным настройкам, а не только к числу копий. Таким образом, сокращается количество ошибок.

Суммируя, можно сказать, что система сама может узнать большинство из тех сведений, которые она запрашивает у пользователя. Главными источниками этих сведений являются:

- здравый смысл разработчика системы,
- предыдущие установленные параметры,
- наиболее часто устанавливаемые параметры.

Единственная проблема этого метода заключается в том, что для его использования к проектированию системы нужно подходить значительно более творчески и тщательно, нежели обычно практикуется.

## **Уровни ошибок и обратная связь**

Помимо классификации человеческих ошибок, приведенной в начале главы, существует ещё одна классификация. В этой классификации ошибки расставлены по уровням их негативного эффекта:

1 Ошибки, исправляемые во время совершения действия, например пользователь перетаскивает файл в корзину и во время перетаскивания замечает, что он пытается стереть не тот файл.

2 Ошибки, исправляемые после выполнения действия, например после ошибочного удаления файла его копия переносится из корзины.

3 Ошибки, которые исправить можно, но с трудом, например реальное стирание файла, при котором никаких его копий не остается.

4 Ошибки, которые на практике невозможно исправить, т. е. ошибки, которые невозможно обнаружить формальной проверкой (т. е. невозможно обнаружить их случайно). Пример: смысловая ошибка в тексте, удовлетворяющая правилам языка.



Каждый хороший программист, умеющий мыслить системно, знает, что ошибок из четвертого пункта нужно всеми силами избегать, не считаясь с потерями, поскольку каждая такая ошибка обходится гораздо дороже, чем любая ошибка из пункта третьего.

Например, межпланетные зонды, из-за ошибок в ПО улетают не туда куда надо, коммерческие договоры, в которых обнаруживаются ошибки, приносят много неприятностей, ошибочные номера телефонов в записной книжке не дают возможности найти абонента – всё это примеры неисправляемых ошибок. Разумеется, такие ошибки всегда обнаруживаются, проблема в том, что к моменту их обнаружения становится поздно их исправлять. Именно поэтому такие ошибки гораздо хуже ошибок, которые исправить трудно, но которые, по крайней мере, сразу видны.

Это было знание программистов. Но не каждый хороший дизайнер интерфейса, знает, что ошибок из второго пункта нужно всеми силами избегать, поскольку каждая такая ошибка обходится гораздо дороже, чем любая ошибка из первого пункта. Объясняется это просто: дизайн интерфейса гораздо моложе программирования.

Вообще говоря, объяснение этого факта двояко, как субъективное, так и объективное.

Объективное объяснение просто: ошибки, исправляемые после, снижают производительность работы. Как мы уже знаем из предыдущей главы, любое действие пользователя состоит из семи шагов. Всякий раз, когда пользователь обнаруживает, что он совершает ошибку, ему приходится возвращаться назад на несколько этапов. Более того, чтобы исправить совершенную ошибку, от пользователя требуется:

- Понять, что ошибка совершена,
- Понять, как её исправить,
- Потратить время на исправление ошибки.

В результате значительный процент времени уходит не на действие (т.е. на продуктивную работу), а на исправление ошибок.

Субъективное объяснение ещё проще: ошибки, исправляемые после, воспринимаются пользователем как ошибки. Ошибки же, исправляемые во время, как ошибки не воспринимаются, просто потому, что для пользователей это не ошибки вообще. Все человеческие действия до конца не алгоритмизированы, они формируются внешней средой (так не получилось и так не получилось, а вот так получилось). Ошибка же, не воспринимаемая как таковая, пользователей не раздражает, что весьма положительно действует на их субъективное удовлетворение от системы.

***Наличие человеческих ошибок, которых нельзя обнаружить и исправить до окончательного совершения действия, всегда свидетельствует о недостаточно хорошем дизайне.***

Теперь пора сказать, как избавиться от ошибок, исправляемых после. Чтобы дать пользователям исправлять их действия на ходу, надо дать им обратную связь.

Когда пользователь совершает какое-то действие – правильное, неправильное или непонятное – интерфейс должен сообщать ему об этом. Нужно всегда уведомлять пользователя о произведенных операциях, изменениях текущего состояния, ошибках или исключениях. О том, привели ли действия пользователя к желаемому результату, могут рассказать визуальные или текстовые сообщения. В интерфейсе Bantam Live (рис. 18) для всех действий существуют линейки-индикаторы степени выполнения.



Рисунок 18

К сожалению, вводить в систему обратную связь получается не всегда. Дело в том, что её не любят программисты. Мотивируют они своё отношение тем, что она плохо влияет на производительность системы. Так что если вы чувствуете, что программисты правы, вспомните, что производительность связки «система-пользователь» всегда важнее производительности системы просто.

Если же и это не помогает, попробуйте спроектировать обратную связь иначе, более скромно. Например, с помощью ползунков на линейке в MS Word можно менять абзацные отступы, при этом обратная связь есть, но не полная: вместо перманентного переформатирования документа по экрану двигается полоска, показывающая, куда передвинется текст.

## **Скорость обучения**

В традиционной науке о человеко-машинном взаимодействии роль обучения операторов чрезвычайно велика. Если человек будет сочтен неподходящим, к системе его просто не допустят.

Напротив, с ПО и сайтами ситуация принципиально иная: как цель ставится возможность работы с системой для любого человека, независимо от его свойств и навыков, при этом целенаправленное обучение пользователей, как правило, не производится. Всё это делает проблему обучения пользователей работе с компьютерной системой чрезвычайно важной.

Почему пользователи учатся?

Есть непреложный закон природы: *люди делают что-либо только при наличии стимула, при этом тяжесть действия пропорциональна силе стимула*. Применительно к компьютерным системам этот закон действует без каких-либо исключений.

Обучение есть действие: если обучаться легко, пользователям будет достаточно слабого стимула, если тяжело, стимул придется увеличивать. Но если стимул для пилота самолета получают преимущественно командными методами (если он не научится определенному минимуму, его просто не допустят до работы), то в случаях компьютерных систем стимул есть вещь почти исключительно добровольная. Это значит, что пользователь обучится пользоваться программой или сайтом только в том случае, если он будет уверен, что это, к примеру, сделает его жизнь легче и приятней.

Пользователь будет учиться какой-либо функции, только если он знает о её существовании, поскольку, не обладая этим знанием, он не способен узнать, что за её использование жизнь даст ему награду. Т.е. одного стимула недостаточно, если пользователь не знает, за что этот стимул дается.

Рассчитывайте на средних пользователей, а не новичков или на профессионалов: средних пользователей, как-никак, абсолютное большинство.

## **Средства обучения**

Обычно считается, что в случае ПО, есть два способа повысить эффективность обучения (помимо метода «обучения плаванию посредством выбрасывания из лодки»), а именно бумажная документация и «оперативная справка». Но существуют более эффективные способы. Рассмотрим некоторые из них:

- Общая «понятность» системы
- Обучающие материалы.

**Понятность системы.** Термин «понятность» включает в себя три составляющих, а именно ментальную модель, метафору, аффорданс и стандарт

**Ментальная модель.** Зачастую, или, точнее, почти всегда, чтобы успешно пользоваться какой-либо системой, человеку необходимо однозначно понимать, как система работает. При этом необязательно точно понимать сущность происходящих в системе процессов, более того, необязательно правильно их понимать. Это понимание сущности системы называется ментальной моделью.

Ментальные модели мы создаем для упрощения картины мира. Наблюдая за событиями, мы их обобщаем, и храним в памяти единую картину. Утюгом никогда не сможет воспользоваться человек, который не знает, что провод от утюга надо воткнуть в розетку. Но, обладая таким знанием, человек может пользоваться утюгом, не зная, сколько энергии утюг потребляет (отсутствие точности), равно как сохраняя искреннюю уверенность, что по проводам, как вода, течёт электричество (отсутствие правильности).

Беда приходит тогда, когда представления человека о системе концептуально не совпадают с реальным устройством системы. Так, например, человек, привезенный на машине времени из прошлого и никогда не встречавшийся с электричеством, поставит утюг на плиту, отчего прибор непременно сгорит.

**Метафора.** Разработать пользовательский интерфейс, в котором модель программы соответствует модели пользователя – задача не из легких. Иногда у пользователей просто нет конкретного представления о том, как работает программа и для чего она предназначена. В таком случае вам придется найти способ подсказать им, как функционирует ваша программа. В графических интерфейсах используется метод метафор. Он позволяет пользователю не создавать новую модель, а воспользоваться готовой моделью, которую он ранее построил по другому поводу. Самая известная метафора, применяемая и в Windows и в Macintosh – это метафора "десктоп" (рабочий стол). Перед вами маленькие папочки с листочками-файлами внутри, последние можно перемещать из одной папки – в другую. Метафора работает, потому что изображения папок напоминают реальные папки, которые мы используем для хранения и сортировки документов в своих кабинетах.

Еще один пример метафоры в интерфейсе – устройство программ для проигрывания звуков на компьютере (рис. 19). Исторически сложилось, что вся аудиотехника имеет почти одинаковый набор кнопок: несколько кнопок со стрелками (назад/вперед), кнопка с треугольником (воспроизведение), кнопка с двумя дощечками (пауза), кнопка с квадратиком (полная остановка) и красный кружок (запись). Про них нельзя сказать, что они совершенно понятны, но научиться им можно без труда. При этом обычно жизнь складывается так, что сначала человек научается пользоваться этими кнопками на материальных устройствах, а уж потом начинает пользоваться компьютером. Соответственно, при проектировании программы аналогичного назначения разумно скопировать существующую систему маркировки кнопок. Благодаря этому пользователям для использования программы ничему не приходится учиться (и даже не приходится переучиваться, что вдвойне обидно, поскольку полностью отрицает возвращение инвестиций в обучение).

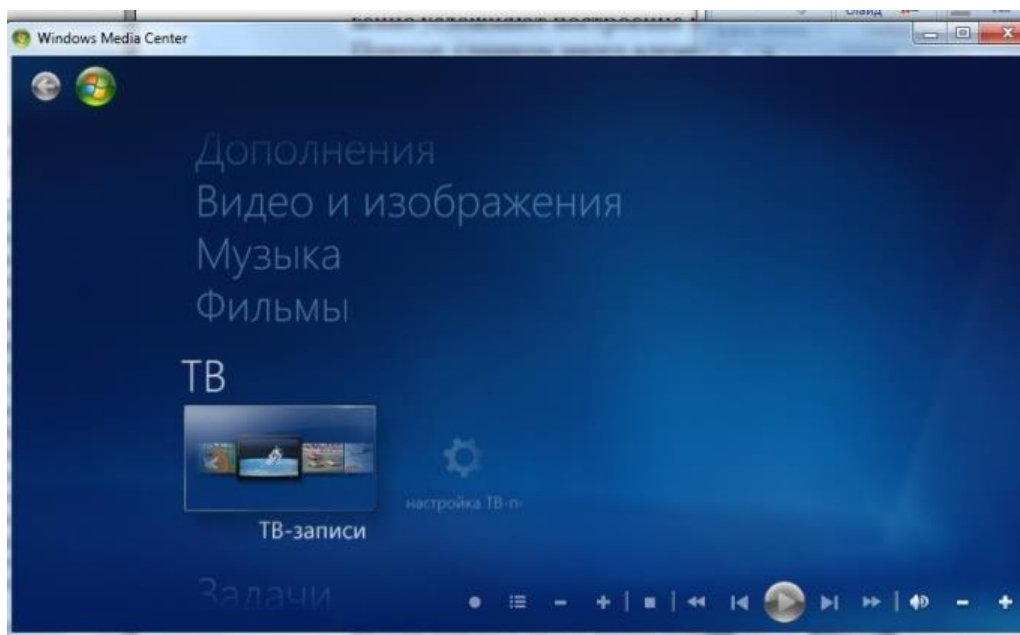


Рисунок 19

Перед вами изображение Kai's Photo Soap (рис. 20). Вы догадаетесь, как поближе рассмотреть картинку?



Рисунок 20

Очевидно, это не вызовет затруднений. Увеличительное стекло – полновесная метафора из реального мира. Вы не опасаетесь, что приближение изменит размер картинки, потому что знаете, что увеличительное стекло сделать этого не может.

Интерфейс Word 2003 содержит два маленьких увеличительных стекла (рис.21), одно из которых (по непонятной причине) фигурирует под названием «Предварительный просмотр», второе же названо «Схема документа» (что это может значить, остается для меня загадкой).

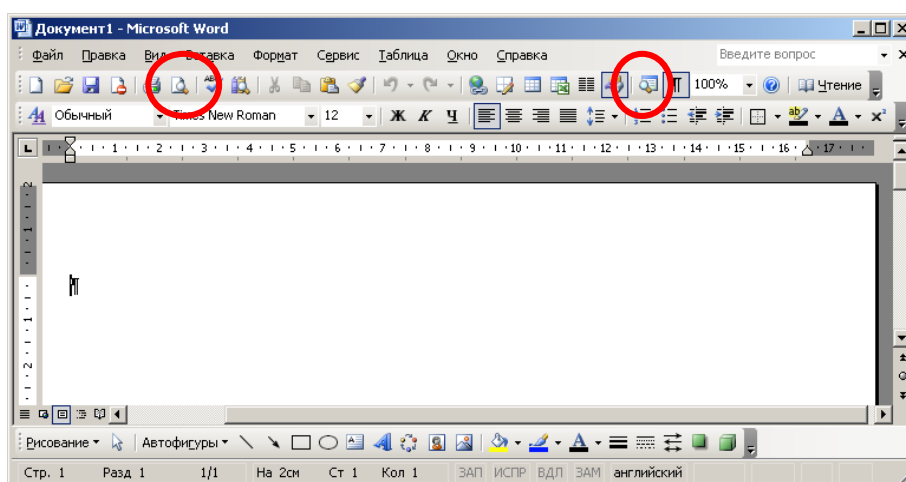


Рисунок 21



На самом деле, увеличить/уменьшить размер документа можно с помощью выпадающего списка, который на данный момент показывает «100%». Метафоры нет, и потому угадать, где скрывается функция приблизить/отдалить, сложнее.

Анализируя опыт применения метафор, можно вывести следующие правила:

- Опасно полностью копировать метафору, достаточно взять из неё самое лучшее;
- Не обязательно брать метафору из реального мира, её смело можно придумать самому;
- Эффективнее всего метафорически объяснять значение отдельных объектов: например, для графической программы слои можно представлять, как положенные друг на друга листы стекла (этот пример подходит и для предыдущего пункта);
- Если метафора хоть как-то ограничивает систему, от неё необходимо немедленно отказаться.

**Аффорданс.** В современном значении этого термина аффордансом называется ситуация, при котором объект показывает субъекту способ своего использования своими неотъемлемыми свойствами.

Например, у двери с ручкой (рис. 22) аффорданс к тому, чтобы ее тянули. Но иногда, ручка бывает у двери, которая открывается наружу, соответственно которую надо толкать. Образуется конфликт между аффордансом ручки (тянуть) и функцией двери (открываться толканием). Чтобы нивелировать этот конфликт зачастую вешают табличку с указанием к действию. Более эффективное и простое решение — убрать ручку. Так образуется аффорданс к толканию.

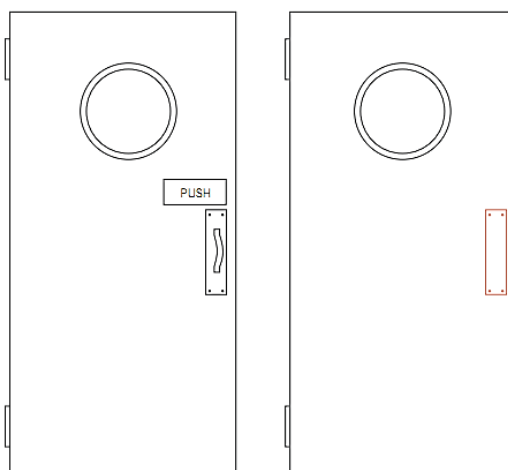


Рисунок 22

Надпись «*На себя*» на двери не является аффордансом, а облик двери, который подсказывает человеку, что она открывается на себя, несет в себе аффорданс.

Еще один пример аффорданса — кирпичи Лего.

По дизайну поверхностей Лего можно легко понять, как кирпичики составляются в фигуры.

Польза аффорданса заключается в том, что он позволяет пользователям обходиться без какого-либо предварительного обучения, благодаря этому аффорданс является самым эффективным и надежным средством обеспечения понятности.

Принцип аффорданса активно используется в компьютерных интерфейсах. Например, выпуклая кнопка сигнализирует нам о том, что ее можно нажать. Дизайн корзины и папок на Рабочем столе, ссылаясь на реальные объекты из жизни, дают понять, как их правильно использовать.

Способы передачи аффорданса:

- Повторение конфигурации объектов конфигурацией элементов управления (этот способ работает хорошо в реальном мире, но не очень хорошо на экране, поскольку предпочтительней непосредственное манипулирование);

- Видимая принадлежность управляющих элементов объекту;

- Визуальное совпадение аффордансов экранных объектов с такими же аффордансами объектов реального мира (кнопка в реальном мире предлагает пользователю нажать на неё, псевдотрехмерная кнопка предлагает нажать на неё по аналогии);

- Изменение свойств объекта при подведении к нему курсора (бледный аналог тактильного исследования).

**Стандарт.** Наконец, остался последний, самый мощный, но зато и самый ненадежный способ обучения, а именно стандарт.

Если что-либо нельзя сделать «самопроизвольно» понятным, всегда можно сделать это везде одинаково, чтобы пользователи обучались только один раз.

Например, кран с горячей водой всегда маркируют красным цветом, а кран с холодной — синим. Частично это соответствует свойствам человеческого восприятия (недаром красный цвет мы называем тёплым, а синий — холодным), но в основном здесь работает привычка.

Основные программы пакета Microsoft Office — Word и Excel — разрабатывались с нуля программистами компании. Другие же были куплены на стороне: FrontPage, например, у Vermeer, или Visio — у Visio. Что у этих программ общего? Дизайн обеих создавался с самого

начала так, чтобы они выглядели и работали как приложения Microsoft Office.

Решение имитировать дизайн приложений Microsoft Office, было принято не просто ради того, чтобы подлизаться к Microsoft. И даже не для того, чтобы впоследствии продать свою программу гиганту. На самом деле, создатель FrontPage Чарльз Фергюссон не скрывает своей неприязни к Microsoft, более того, он неоднократно призывал Департамент юстиции США принять хоть какие-нибудь меры по отношению к «Редмонским бестиям» (до тех пор, пока не продал им свою компанию). Vermeer и Visio скопировали пользовательский интерфейс Microsoft Office, должно быть, просто оттого, что это было выгодно: быстрее и проще, чем изобретать велосипед.

Когда менеджер отдела программирования Microsoft Майк Матье загрузил FrontPage с веб-страницы Vermeer, оказалось, что программа работает во многом так же как и Word. Поскольку она работала в соответствии с его ожиданиями, пользоваться ею было просто. И эта простота в использовании программы в одночасье произвела самое благоприятное впечатление на господина Матье.

Дизайн элементов управления, выдержанный в едином ключе для различных программ, помогает пользователю обучиться работать с новой программой.

Однако, стандарт штука мощная, но зато ненадежная. Он очень хорошо работает, когда популярен, в противном случае не работает вовсе.

Популярность стандарта может быть достигнута двумя способами: во-первых, он может быть во всех системах, во-вторых, он может быть популярен внутри отдельной системы.

Например, стандарт интерфейса MS Windows популярен почти во всех программах для Windows, именно поэтому его нужно придерживаться. С другой стороны, этот стандарт оставляет неопределенным очень многое (никто да не обнимет необъятного), и это многое в разных системах трактуется по-разному.

***Последовательность в реализации интерфейса есть первое условие качества результата.***

**Обучающие материалы.** Количество подсистем справки, нужных для того, чтобы пользователь научился пользоваться системой, довольно невелико, так что все их можно легко разобрать. Под «подсистема справки» мы будем иметь в виду часть справочной системы, которая выполняет сугубо определенные функции и требует сугубо определенных методов представления.

**Базовая справка** – объясняет пользователю сущность и назначение системы. Обычно должна сработать только один раз. Как правило, не требуется для ПО, зато почти всегда требуется для сайтов.

**Обзорная справка** – рекламирует пользователю функции системы. Также обычно срабатывает один раз. Нужна и ПО и сайтам. Поскольку у зрелых систем функциональность обычно очень велика, невозможно добиться того, чтобы пользователи запоминали её за один раз. В этом случае оптимальным вариантом является слежение за действиями пользователя и показ коротких реклам типа «*А вы знаете, что...*» в случае заранее определенных действий пользователей (примером такого подхода являются помощники в последних версиях MS Office).

Обычно базовая и обзорная справки реализуются в бумажной документации. Однако в последнее время появилась возможность интегрировать в справочную систему видео при помощи либо Macromedia Flash, либо Shockwave. Нет сомнений, что реклама, поданная в виде анимации, способна как повысить желание её просмотреть, так и повысить субъективное удовлетворение пользователей от системы. Это также создаёт желание у пользователя учиться.

**Справка предметной области** – отвечает на вопрос «*Как сделать хорошо?*». Поскольку от пользователей зачастую нельзя рассчитывать знания предметной области, необходимо снабжать их этим знанием на ходу.

При этом действуют два правила:

- Во-первых, пользователи не любят признавать, что они чего-либо не знают, соответственно, подавать это знание надо максимально «небрежным тоном»;
- Во-вторых, наличие такого знания всегда повышает субъективную оценку справочной системы в целом, т.е. приводит к тому, что пользователи чаще обращаются к справочной системе и от этого эффективней учатся.

Справка предметной области также реализуется обычно в бумажной документации. Однако, как минимум, часть её можно подавать пользователям в интерфейсе вместе с выдержками из обзорной справки (лучше динамически, *a la* «Помощник» из MS Office).

Справка предметной области является самой важной подсистемой справки. Но без знания предметной области он никогда не сможет пользоваться системой правильно и эффективно.

**Процедурная справка** – отвечает на вопрос «*Как это сделать?*». В идеале она должна быть максимально более доступна, поскольку

если пользователь не найдет нужную информацию быстро, он перестанет искать и так и не научится пользоваться функцией (возможно, никогда). Разработчики чаще всего не привязывают темы справки к интерфейсу: когда пользователям непонятно, как выполнить нужное им действие, им приходится искать в справочной системе нужную тему. Это неправильно, тем более что технических проблем в этом нет.

**Контекстная справка** – отвечает на вопросы «*Что это делает?*» и «*Зачем это нужно?*». Как правило, наибольший интерес в ПО представляет первый вопрос, поскольку уже по названию элемента должно быть понятно его назначение (в противном случае его лучше вообще выкинуть), а в Интернете – второй (из-за невозможности предугадать, что именно будет на следующей странице).

Поскольку пользователи обращаются к контекстной справке во время выполнения какого-либо действия, она ни в коем случае не должна прерывать это действие (чтобы не ломать контекст действий), её облик должен быть максимально сдержанным, а объем информации в ней – минимальным.

Для контекстной справки заслуженно используют всплывающие подсказки (*ToolTip*) и, в последнее время, «пузыри».

Всплывающие подсказки хорошо справляются с ответом на вопросы «*Что это такое?*» и «*Зачем это нужно?*», при условии, что объем ответов сравнительно невелик. Они не занимают пространства экрана и не отвлекают внимания пользователей.

Если разработчики ПО уже привыкли писать ко всем объектам и элементам управления подсказки, то для веб-дизайнеров это пока экзотика. Интересно при этом, что в интернете контекстная справочная система, как правило, более нужна, нежели в ПО – просто потому, что большинство сайтов являются однократно используемыми системами, пользователями которых являются изначально необученные люди.

**Справка состояния** – отвечает на вопрос «*Что происходит в настоящий момент?*». Поскольку она требуется именно что в настоящий момент, она не может быть вынесена из интерфейса. В целом это самая неproblemатичная для разработчиков система справки, так что в этой книге разбираться она не будет.

**Система должна индексировать все свои состояния.**

**Спиральность.** Поскольку пользователи обращаются к справочной системе при возникновении проблем, можно смело сказать, что использование справочной системы всегда воспринимается негативно. Поэтому следует сокращать объем справочной системы, чтобы сократить длительность неудовольствия. Однако при малом объеме спра-

вочной системы возрастает риск того, что пользователи не найдут в ней ответы на свои вопросы.

Эффективный метод решения этой проблемы: так называемые **спиральные тексты**. Идея заключается в следующем. При возникновении вопроса пользователь получает только чрезвычайно сжатый, но ограниченный ответ (1-3 предложения). Если ответ достаточен, пользователь волен вернуться к выполнению текущей задачи, тем самым длительность доступа к справочной системе (и неудовольствие) оказывается минимальной. Если ответ не удовлетворяет пользователя, пользователь может запросить более полный, но и более объемный ответ. Если и этот ответ недостаточен (что случается, разумеется, весьма редко), пользователь может обратиться к ещё более подробному ответу.

Таким образом, при использовании этого метода, пользователи получают именно тот объем справочной системы, который им нужен. Спиральность текста считается нормой при разработке документации. Есть веские основания считать, что она необходима вообще в любой справочной системе. Учитывая тот факт, что разработка спирали в справке не проблематична, рекомендуется делать её во всех случаях.

### **Субъективное удовлетворение пользователей**

Натан Мирвольд, бывший вице-президент Microsoft, некогда высказал скандальное по тем временам изречение «*Cool is a powerful reason to spend money*» (Слово «Cool», к сожалению, плохо переводится на русский язык. Это изречение интересно, прежде всего, тем, что представляет собой просто триумф субъективности).

Предположение Мирвольда оправдалось. Субъективные факторы имеют тот же вес, что и объективные. Субъективность доминирует над объективностью только в тех случаях, когда покупателем системы выступает сам пользователь.

Но и в прочих случаях роль «крутоты» зачастую существенна, хотя бы потому, что повышение количества радости при прочих равных почти всегда приводит к повышению человеческой производительности. Это делает неактуальными вечные споры о первичности формы или функции. И то и другое важно.

Какие же факторы влияют на субъективное удовлетворение?

Все факторы связаны с субъективными ощущениями человека. А именно с субъективными ощущениями: эстетики, времени работы, психологического напряжения, собственной глупости, самовыражения.



**Субъективное ощущение эстетики.** Все знают, что значительно легче и приятнее пользоваться эстетически привлекательными объектами. Это наблюдение породило весь промышленный дизайн, включая дизайн одежды, интерьеров и так далее.

В то же время в дизайне интерфейсов, это наблюдение до сих пор как следует, не утвердилось: бои между пуристами (интерфейс должен быть, прежде всего, работоспособным) и маньеристами (красота – это страшная сила) никоим образом не затихают.

В то же время «срединный путь» до сих пор не найден, интерфейсы, равно удобные и эстетически привлекательные, до сих пор существуют в единичных экземплярах. Эстетическая привлекательность не измеряет красоту приложения и не характеризует его стиль; скорее, она представляет, насколько хорошо внешний вид и поведение приложения характеризует его функции.

Программисты заботятся о функциональности приложения, однако внешний вид приложения может влиять на подсознательное поведение пользователей при работе с приложением. Поэтому, в приложениях, которые помогают пользователям выполнить серьезную задачу, можно поставить акцент на задаче, сохраняя декоративные элементы тонкими и ненавязчивыми, и управлять ими с помощью стандартных средств управления предсказуемого поведения. Эти приложения посылают четкие, унифицированные сообщения о своей цели и своей идентичности, которая помогает людям использовать их.

Если приложение, решая серьезные задачи, выглядит навязчивым и легкомысленным, то пользователи могут подвергнуть сомнению надежность приложения. С другой стороны, в приложениях, таких как игра – пользователи ожидают увидеть красочный внешний вид, который обещает получение удовольствия и радости от игры. И это вызывает желание открыть это приложение. Люди не ожидают серьезного внешнего вида от игры. Внешний вид игры и поведение должны соответствовать ее назначению.

Итак, какие их принципы могут быть использованы в дизайне интерфейса?

Для того, чтобы интерфейс был эстетически привлекательным, необходимо, чтобы он был незаметен в процессе его использования.

Во что бы то ни стало, добивайтесь того, чтобы интерфейс был неощущаемый. Для этого:

- Избегайте развязности в изображении Лучше, чтобы он был скромнее.

• Избегайте ярких цветов. Существует очень немного цветов, обладающих и яркостью, и мягкостью (т.е. не бьющих по глазам). На экране их значительно меньше, поскольку в жизни такие цвета обычно моделируются как собственно цветом, так и текстурой, с чем на экране есть проблемы.

• Избегайте острых углов в изображении.

• Старайтесь сделать изображение максимально более легким и воздушным.

• Старайтесь добиваться контраста не сменой насыщенности элементов, а расположением пустот.

• Старайтесь минимизировать количество констант (тем более, что двух констант обычно хватает на все). Разумеется, единожды примененных закономерностей необходимо придерживаться во всей системе.

***Стремитесь не столько к красоте интерфейса, сколько к его элегантности.***

Красота понятие относительное. Для одних красивыми могут считаться только живописные закаты, для других картины художника Кустодиева, а для третьих – комбинация вареных сосисок, зеленого горошка и запотевшей бутылки пива. Это делает красоту вещь не слишком универсальной. Хуже того. Любая красота со временем надоедает и в лучшем случае перестает восприниматься.

Лучший интерфейс должен соблюдать баланс между ясностью целей и простотой использования. Пытаясь создать оригинальный продукт, проектировщики часто злоупотребляют «украшательством».



Рисунок 23

Например, представьте себе приложение калькулятор (рис. 23), которое использует сложный, художественный стиль и об разное расположение, чтобы отобразить знакомые элементы калькулятора. Пользователи, конечно, могут понять, как нажать кнопки и прочесть результат.

Но для людей, которым просто необходим калькулятор для выполнения своей работы, новизна интерфейса стирается быстро и красивый пользовательский интерфейс становится помехой.

В отличие от этого, GarageBand мог бы помочь людям создавать музыку без отоб-

ражения красивых, реалистичных инструментов (рис. 24), но это сделало бы приложение менее интуитивным, и менее приятным в использовании.

В GarageBand, пользовательский интерфейс не только показывает людям, как использовать приложение, но и делает основную задачу (создание музыки) – проще.

Именно поэтому в интерфейсах обычно не место красоте.



Рисунок 24

Элегантность и гармония гораздо лучше. Во-первых, они не надоедают. Во-вторых, они редко осознаются потребителями, обеспечивая неощущаемость. В-третьих – они приносят эстетическое удовольствие независимо от культурного уровня потребителя (так, древнегреческие и слегка менее древние римские здания воспринимаются нами красивыми, несмотря на абсолютную разницу культур и времени). В-четвертых, в производстве они гораздо удобнее красоты, поскольку сравнительно легко ставятся на поток.

Итак, каким образом надо действовать, чтобы добиться элегантности:

- Старайтесь сделать интерфейс максимально насыщенным визуальными закономерностями (рис. 25). Есть универсальное правило – чем больше закономерностей, тем больше гармонии. Даже самые незначительные закономерности всё равно воспринимаются. Под закономерностью понимают любое методически выдерживаемое соответствие свойств у разных объектов, например, высота кнопок может быть равна удвоенному значению полей диалогового окна.

- Всемерно старайтесь использовать модульные сетки, т.е. привязывайте все объекты к линиям (лучше узлам) воображаемой сетки, которую выдерживайте во всем интерфейсе.

- Старайтесь привязывать все размеры и координаты (как минимум пропорции диалоговых окон) к золотому сечению (0.618 x 0.382).

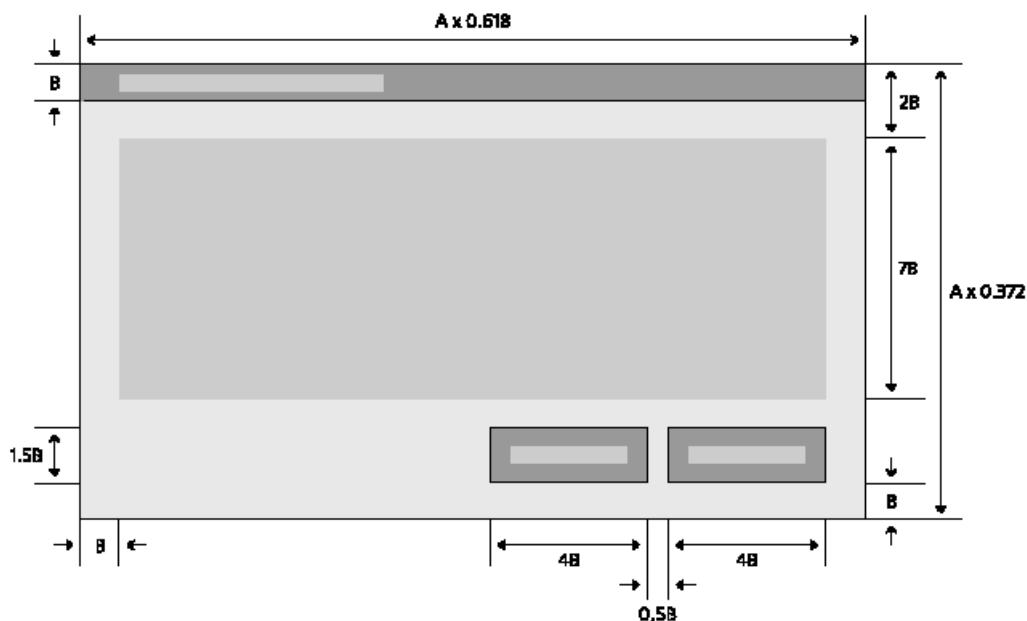


Рисунок 25

**Субъективное ощущение времени.** Любой человек хочет работать быстро. Если работу можно выполнить быстро, у человека возникает приятное ощущение. Однако субъективное ощущение времени зачастую сильно отличается от объективного, так что методы повышения реальной скорости работы, о которых мы говорили вначале лекции, помогают отнюдь не всегда.

Воспринимаемая продолжительность действий напрямую зависит от уровня активности пользователя. Все вы знаете, что при бездействии (скуке) время течет невыносимо медленно.

Действие может быть не обязательно физическим: лежа на диване и смотря фильм, т.е. не совершая почти никаких физических действий, время можно потратить очень быстро.

Таким образом, субъективную скорость работы можно повысить двумя способами:

- Заполнение пауз между событиями. Есть данные о том, что если в периоды ожидания реакции системы пользователям показывается индикатор степени выполнения, субъективная продолжительность паузы существенно снижается. Судя по всему, чем больше информации предъявляется пользователям в паузах, тем меньше субъективное время. С другой стороны, эта информация может вызвать стресс в кратковременной памяти, так что пользоваться этим методом надо осторожно.

- Разделение крупных действий пользователей на более мелкие. При этом количество работы увеличивается, но зато субъективная

длительность снижается. Плох этот метод тем, что увеличивает усталость.

С другой стороны, повышение объективной скорости работы зачастую способно повысить и субъективную скорость.

**Субъективное ощущение психологического напряжения.** Пользователь знает, что во время работы может что-либо испортить. Он может отформатировать жесткий диск, может стереть или испортить нужный файл. И это вызывает у него психологическое напряжение, иначе говоря – стресс.

Пользователь, зная, что он не может совершить ошибку, испытывает радость и умиротворение. Чтобы добиться этого, необходимо иметь возможность:

- Отмены пользователем своих предыдущих действий, без ограничения количества уровней отмены и типа отменяемых действий. Задача эта непростая, но зато результат крайне существенен. К сожалению, создание таких систем требует смены модели мышления программистов. Зачастую более реалистичным решением является давно уже существующая практика:

- Прятать опасные для пользователя места интерфейса. Проблема заключается в том, что при этом логично прятать все функции, изменяющие данные, например банальная функция автоматической замены, может мгновенно уничтожить текст документа.

Другим фактором, существенно влияющим на субъективное удовлетворение пользователей, является чувство контроля над системой.

Существует значительная часть пользователей, для которой использование компьютера не является действием привычным. Для таких пользователей ощущение того, что они не способны контролировать работу компьютера, является сильнейшим источником стресса.

Таким образом, пользователей нужно всемерно снабжать ощущением, что ничего не может произойти, пока этого не захочется самому пользователю. Функции, работающие в автоматическом режиме, но время от времени просыпающиеся и требующие от пользователей реакции, вызывают стресс. В любом случае, стоит всеми силами внушать пользователям мысль, что только явно выраженное действие приводит к ответному действию системы (это, в частности, главный аргумент против ролловеров – пользователь ещё ничего не нажал, а уже что-то произошло).

**Субъективное ощущение собственной глупости.** Ни один пользователь не может долго и продуктивно работать с системой, которая говорит, что он глуп. Тем не менее, такие «скандальные» системы яв-

ляются нормой. Виной тому сообщения об ошибках. Почему сообщения об ошибках плохи?

Дело в том, что большинство сообщений об ошибках в действительности не являются собственно сообщениями об ошибках. На самом деле они показывают пользователю, что система, которой он пользуется:

- Недостаточно гибка, чтобы приспособиться к его действиям;
- Недостаточно умна, чтобы показать ему его возможные границы его действия;
- Самоуверенна и считает, что пользователь дурак, которым можно и нужно помыкать.

Разберем это подробнее.

**Недостаточная гибкость.** Многие программы выдают сообщения об ошибке, когда пользователь хочет с их точки зрения невозможного. Вот что бывает (рис. 26), если пользователь попытается ввести значение, которое ему нужно, но которое система не умеет обрабатывать.

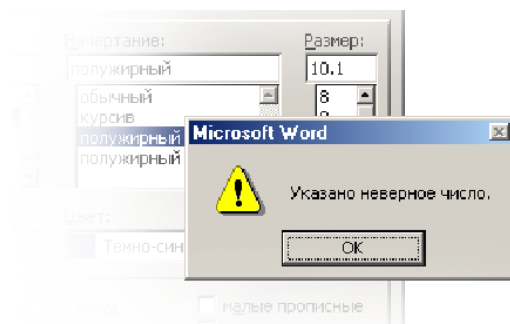


Рисунок 26

Тут возможно три альтернативных решения:

- Во-первых, при проектировании системы можно более тщательно подходить к выбору её функциональности.
- Во-вторых, можно просто игнорировать неправильность значения, округляя его до ближайшего возможного (индицируя, возможно, самостоятельность действий системы однократным миганием соответствующего поля ввода).
- В-третьих, вместо обычного поля ввода можно использовать «крутилку».

В действительности множество действий пользователя направлены не на то, чтобы сделать так, а не иначе, а на то, чтобы сделать примерно так, как хочется. Пользователю часто нет дела, можно добиться точного результата или нет. Показывать ему в таких случаях



диалог об ошибке глупо, поскольку пользователю не нужно ничего знать. Ему нужен результат.

**Нежелание показать границы действия.** Во многих случаях пользователь совершает действия, которые воспринимаются программой как неправильные, не потому, что он дурак, но потому, что система не показала ему границ возможного действия.

Типичное сообщение об ошибке, вызванное нежеланием системы показать пользователю границы его действий (рис. 27).

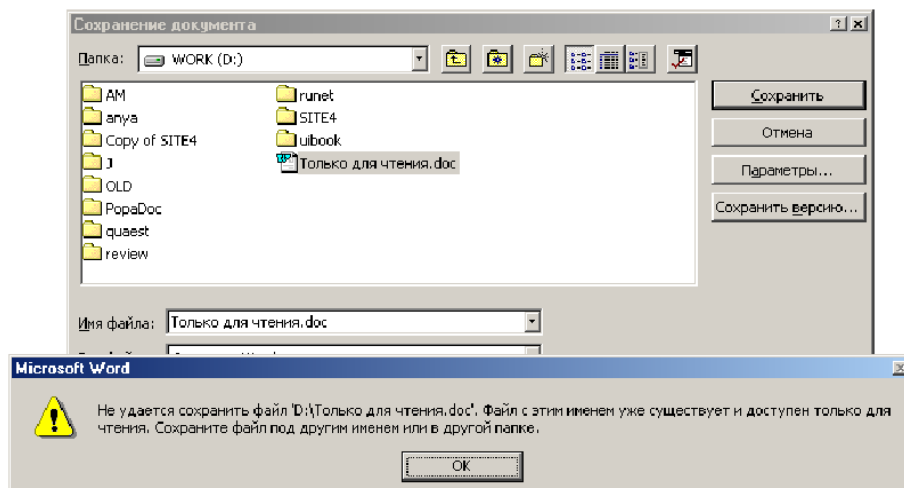


Рисунок 27

С одной стороны, оно разумно – файл не может быть записан под этим именем. С другой стороны, это сообщение показывается пользователю каждый раз при попытке перезаписать такой файл.

Если бы названия всех защищенных от записи файлов отображались бы не черными, а серыми, это сообщение пришлось бы показывать пользователю только один раз в его жизни.

Все такие сообщения, можно было бы избежать, просто блокировав возможность совершения некорректных действий или показав пользователю их некорректность до совершения действия.

**Самоуверенность.** Нормой также являются случаи, когда система пытается выставить дело так, как будто пользователь идиот, а система, наоборот, есть воплощение безошибочности и правоты (рис. 26).

Для кого неверное? И кто, собственно, виноват, система или пользователь? В действительности не пользователь сделан для системы, а система для пользователя. Таким образом, как-либо ущемлять пользователя неправильно.

***Пользователи ненавидят сообщения об ошибках.***

Суммируя, можно сказать, что почти любое сообщение об ошибке есть признак того, что система спроектирована плохо.

Всегда можно сделать так, чтобы показывать сообщение было бы не нужно. Любое сообщение об ошибке говорит пользователю, что он дурак (рис. 28).

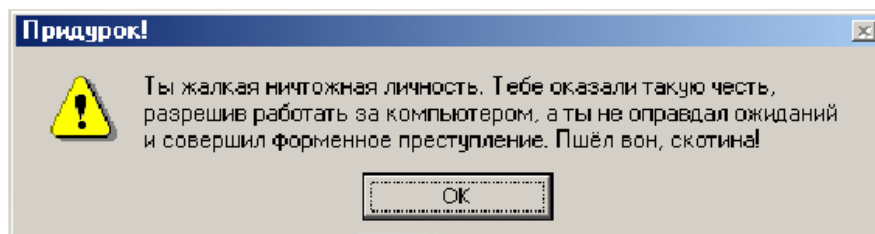


Рисунок 28

Именно так пользователи воспринимают любые сообщения об ошибках. Таким образом, почти все сообщения об ошибках должны быть удалены. Разумеется, речь идет не о том, чтобы просто выкинуть куски кода из программы, а о том, что системы изначально надо проектировать так, чтобы в них отсутствовала необходимость в таких сообщениях. Невозможно полноценно работать с системой, которая по несколько раз за день тебя ругает.

#### **Каким должно быть сообщение об ошибке?**


Многие пренебрегают подбором формулировок для сообщений об ошибках, считая их чем-то вторичным. Разработчики обычно знают, как избегать неправильных действий в своих продуктах, и потому редко сталкиваются с подобными сообщениями.

Но настоящие пользователи не могут этим похвастаться. Если пользователь плохо понимает, что ему делать после возникновения сообщения об ошибке, то он может вообще забросить ваш продукт. Поэтому текст сообщения может быть игривым, но как минимум должен помогать пользователям, быстро разобраться с причиной и избежать сбоев в дальнейшем.

Старайтесь избегать специализированных терминов, пишите сообщения общедоступным, понятным языком. Не надо вываливать пользователю информацию, которой он не может воспользоваться. При желании даже ошибки проверки можно перефразировать, чтобы они звучали более дружелюбно.

С помощью сообщений об ошибках можно сразу выделить конкретные поля ввода, на которые нужно обратить внимание пользователя. Иногда сообщения об ошибках такие бестолковые, что не помо-

гают пользователю решить проблему. Всегда старайтесь сразу визуально подсказать, где именно пользователь должен решить возникшее затруднение. На рисунке 29 представлена форма регистрации Basecamp, в которой не только подсвечиваются нужные поля, но и даётся текстовая подсказка по заполнению.

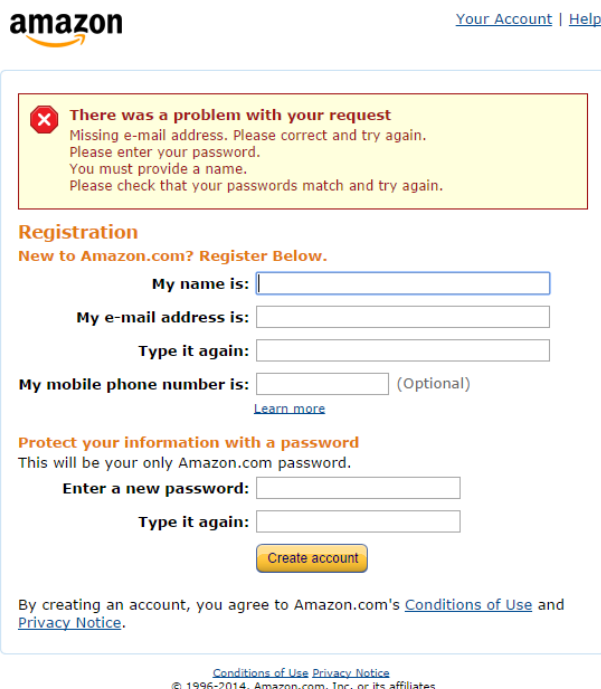


Registration form for Basecamp:

- Your full name:  (highlighted with a red border and error message: "Enter your full name")
- Company or organization:
- Email:  (highlighted with a red border and error message: "Enter a valid email")
- Password:  (highlighted with a red border and error message: "Enter a password")
- Submit button: "Start my two month free trial"

Рисунок 29

Вероятно, вы считаете, что если в форме всего два-три поля, то всё настолько очевидно, что пользователю помогать не нужно. Однако даже в этом случае рекомендуется фокусировать внимание людей на проблемных местах, это куда полезнее, чем просто вываливать им список ошибок, как это делается в Amazon (рис. 30).



Amazon registration form with error messages:

- Amazon logo and links: [Your Account](#) | [Help](#)
- Error message box: "There was a problem with your request. Missing e-mail address. Please correct and try again. Please enter your password. You must provide a name. Please check that your passwords match and try again."
- Registration section: "New to Amazon.com? Register Below."
- Form fields: "My name is:", "My e-mail address is:", "Type it again:", "My mobile phone number is:" (Optional), "Learn more"
- Password section: "Protect your information with a password. This will be your only Amazon.com password. Enter a new password:", "Type it again:", "Create account"
- Footer: "By creating an account, you agree to Amazon.com's [Conditions of Use](#) and [Privacy Notice](#). © 1996-2014, Amazon.com, Inc. or its affiliates"

Рисунок 30

Теперь можно рассказать, каким должно быть сообщение об ошибке, тем более, что ничего сложного в создании идеального сообщения нет. Напротив, всё очень просто. Идеальное сообщение об ошибке должно отвечать всего на три вопроса:

- В чем заключается проблема?
- Как исправить эту проблему сейчас?
- Как сделать так, чтобы проблема не повторилась?

**Самовыражение.** Страсть к самовыражению является одной из самых сильных черт человеческой природы. В большинстве случаев люди самовыражаются через вещи (двигают мебель и покупают модную одежду), когда нет вещей – насвистывают или поют изящные песни «лишенным приятности голосом».

Ни один человек не может существовать сколько-нибудь продолжительное время в обстановке, не допускающей самовыражения.

Неудивительно, что пользователи хотят выразить себя и в программах, которыми они пользуются. Соответственно, возможность настроить систему под свои нужды является мощной причиной субъективного удовлетворения.

Проблема здесь в том, что это очень дорого стоит. Времени на самовыражение уходит крайне много. По грубым прикидкам, на такой процесс самовыражения в среднем уходит около 45 минут в неделю. Получается, что организация всего с сотней работников теряет на этом еженедельно 75 часов, что почти равняется потере недельной работы двух человек.

С другой стороны, настроенный под свои нужды интерфейс, судя по всему, снижает усталость работников и повышает их рабочее настроение. Таким образом, в продуктах, продаваемых пользователям напрямую, возможность настройки под конкретного пользователя обязательна.

Во всех остальных случаях, судя по всему, нужен способ настройки, позволяющий максимально изменить вид системы минимумом команд (чтобы снизить время, затрачиваемое на самовыражение). Хорошим вариантом является банальный выбор варианта готовой настройки из списка без возможности модифицировать встроенные варианты.

Чтобы пользователь не отвлекал свое внимание от задачи при настройке интерфейса, необходимо соблюдать следующие принципы:

- **Всегда есть причина для настройки.** В идеале, настройки пользовательского интерфейса облегчают выполнение задачи пользователями и увеличивают их опыт. Насколько это, возможно, необходимо предоставлять пользователю возможности настройки.

- **Избегайте, увеличения когнитивной нагрузки на пользователя.** Пользователи знакомы с внешним видом и поведением стандартных элементов пользовательского интерфейса, поэтому они не думают о том, как использовать их. Когда они сталкиваются с элементами, которые выглядят или ведут себя не как стандартные, они не могут использовать свой предыдущий опыт. Если ваши уникальные элементы не делают выполнение задачи проще, то пользователи не будут изучать новые процедуры, которые не использовались в каких-либо других приложениях.

- **Будьте внутренне непротиворечивым.** Чем больше ваш пользовательский интерфейс, тем более важно быть последовательным в представлении внешнего вида и поведения элементов приложения. Для того, чтобы пользователи могли понимать, как использовать незнакомые элементы управления и, чтобы они имели возможность использовать эти знания на всех этапах приложения.

- **Всегда обращать внимание на содержание.** Пользователи в основном знакомы с элементами управления, и поэтому главное внимание надо обращать не на элементы управления, а на контент. Как настроить пользовательский интерфейс, чтобы он не заслонял содержание. Например, при разработке вашего приложения предназначенного для просмотра видео, вы можете выбрать элементы управления воспроизведения. Но используете ли вы уникальные или стандартные элементы управления воспроизведения не так важно, как то исчезают ли эти элементы управления из поля зрения после того, как пользователь начинает просмотр видео, и появляются ли они снова при активации интерфейса.

- **Подумайте дважды, прежде чем перепроектировать стандартный элемент управления.** Если вы планируете делать больше, чем настройки стандартного управления, убедитесь, что ваш переработанный интерфейс управления несет больше информации, чем стандартный. Например, если вы создаете элемент управления переключатель, который не указывает на наличие противоположного значения, люди могут это не понять.

- **Обязательно тщательно тестируйте созданные элементы интерфейса.** Во время тестирования внимательно наблюдайте за пользователем, чтобы увидеть, могут ли они предсказать, что будут делать ваши элементы и смогут ли они легко с ними взаимодействовать. Если, например, вы создаете элемент управления, который имеет размер меньше, чем 44x44 dp, пользователи будут иметь проблемы с его активацией.

## ЛЕКЦИЯ № 6. КОЛИЧЕСТВЕННЫЙ АНАЛИЗ ИНТЕРФЕЙСА

Количественные методы помогают свести спорные вопросы в оценке качества интерфейса к простым вычислениям.

Одним из лучших подходов к количественному анализу моделей интерфейсов является классическая модель *GOMS (the model of goals, objects, methods and selection rules)* – модель целей, объектов, методов и выбор правил.

Эта модель основана на оценке скорости печати. Время, требуемое для выполнения какой-то задачи системой пользователь-компьютер, является суммой всех временных интервалов, которые потребовались системе на выполнение элементарных жестов, составляющих данную задачу. Лабораторным путем установлены стандартные средние интервалы для некоторых жестов, выполняемых различными пользователями: К = 0,2 с – нажатие клавиши; Р = 1,1 с – указание (на какую-то позицию на экране монитора); Н = 0,4 с – перемещение (руки с клавиатуры на «мышь» или обратно; М = 1,35 с – ментальная подготовка – мысленный выбор пользователем своего следующего элементарного действия; R – ответ (время ожидания ответа компьютера).

### Расчет по модели GOMS

Основные правила, позволяющие определить, в какие моменты будут проходить ментальные операции, представлены в табл. 3.

Таблица 3

<b>Правило 0</b> Начальная расстановка операторов <b>М</b>	Оператор <b>М</b> устанавливается перед всеми операторами <b>К</b> (нажатие клавиши) и <b>Р</b> , предназначенными для выбора команд, если <b>Р</b> указывает на аргументы этих команд оператор <b>М</b> не ставится	
<b>Правило 1</b> Удаление ожидаемых операторов <b>М</b>	Если оператор, следующий за оператором <b>М</b> ожидаемый с точки зрения оператора, предшествующего <b>М</b> , то этот оператор <b>М</b> может быть удален и последовательность <b>РМК</b> превращается в <b>РК</b>	



<p><b>Правило 2</b> Удаление операторов <b>М</b> внутри когнитивных единиц</p>	<p>Если строка типа МКММКМК... принадлежит когнитивной единице, то следует удалить все операторы <b>М</b>, кроме первого. Когнитивной единицей является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент.</p>	
<p><b>Правило 3</b> Удаление <b>М</b> перед последовательными разделителями</p>	<p>Если оператор <b>К</b> означает лишний разделитель, стоящий в конце когнитивной единицы (например разделитель команды, следующий сразу за разделителем аргумента этой команды) то следует удалить оператор <b>М</b>, стоящий перед ним.</p>	
<p><b>Правило 4</b> Удаление операторов <b>М</b>, которые являются прерывателями команд</p>	<p>Если оператор <b>К</b> является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор <b>М</b>, стоящий перед ним. Но если оператор <b>К</b> является разделителем для строки аргументов или любой другой изменяемой строки, то <b>М</b> следует сохранить перед ним.</p>	

<b>Правило 5</b> Удаление перекрывающихся операторов <b>М</b>	Любую часть операторов <b>М</b> , которая перекрывает оператор <b>Р</b> , означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.	
--	---	--

**Пример расчета.** Имеется вариант интерфейса пользователя созданный для задачи перевода температуры из шкалы по Цельсию, в шкалу по Фаренгейту и наоборот (рис. 32).

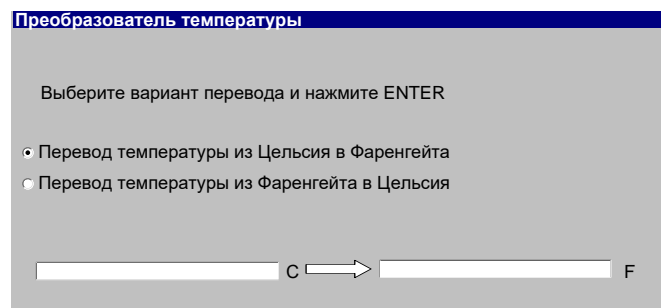


Рисунок 31

Для перевода температуры в данном варианте необходимо выполнить следующие операции:

- Перемещение руки к «мышь» (Н);
- Перемещение курсора к необходимому переключателю в группе (НР);
- Нажатие на необходимый переключатель (НРК);
- Перемещение рук снова к клавиатуре (НРКН);
- Ввод 4-х символов (НРКНKKKK);
- Нажатие клавиши «Enter» (НРКНKKKKK);

Применим **Правило 0** для расстановки индексов **М** и получим выражение:

НМРМКНМКМКМКМКМК.

В соответствии с **Правилом 1** РМК заменим на РК и в соответствии с **Правилом 2** удалим операторы **М** внутри когнитивных единиц. Выражение приобретает вид:

НМРКНМККККМК.

Заменим символы на временные интервалы: К=0,2; Р=1,1; Н=0,4; М=1,35.

Теперь сложим операторы и вычислим сумму времен:

$$\begin{aligned} & H+M+P+K+H+M+K+K+K+K+M+K \\ & 0,4+1,35+1,1+0,2+0,4+1,35+(4 \times 0,2)+1,35+0,2 = 7,15 \text{ с.} \end{aligned} =$$

## Измерение эффективности интерфейса

Если имеется модель интерфейса, то с помощью GOMS можно определить время, необходимое пользователю для выполнения любой, четко сформулированной задачи, для которой данный интерфейс предусмотрен. Однако модели анализа не могут дать ответ на вопрос о том, насколько быстро должен работать интерфейс. Чтобы ответить на него, необходимо воспользоваться мерой, применяемой в теории информации.

Для оценки времени, необходимого для выполнения задачи, следует определить минимально количество информации, которое пользователь должен ввести, чтобы выполнить задачу.

Если методы работы, используемые в предполагаемом интерфейсе, требуют введения такого количества информации, которое превышает минимальное, это означает, что пользователь делает лишнюю работу и поэтому интерфейс следует усовершенствовать.

**Информационная производительность** интерфейса **E** определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь.

Так же как и в отношении физической производительности, параметр **E** может изменяться в пределах от **0** до **1**.

Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет **1** (чтобы избежать деления на 0).

Производительность **E** может равняться и **0** в случаях, когда пользователь должен ввести информацию, которая бесполезна для выполнения задачи.

В параметре **E** учитывается только информация, необходимая для задачи и информация, вводимая пользователем.

Так для выбора одного варианта из четырех возможных достаточно двух бит информации, из восьми – трех, из 16 – 4-х и т.д.

Несколько методов действия могут иметь одинаковую производительность **E**, но иметь разное время выполнения. Возможно, что один метод имеет более высокий показатель **E**, но действует медленнее, чем другой метод.

При количестве **n** равновероятных вариантов суммарное количество передаваемой информации определяется как:

$$\log_2 n$$

Количество информации для каждого варианта определяется как:

$$(1/n) \log_2 n \quad (1)$$

Например, при передаче информации нажатием клавиши ее количество зависит от общего числа клавиш и относительной частоты использования каждой из них.

Если на клавиатуре имеется 128 клавиш, и каждая из них используется с одинаковой частотой, то нажатие любой из них будет передавать:

$$\frac{1}{128} \log_2(128) = 7 \quad \text{бит}$$

Если вероятности для каждого варианта или элемента не являются равными и  $i$ -й вариант имеет вероятность  $P(i)$ , то информация, передаваемая этим вариантом определяется, как:

$$p(i) \log_2 (1/p(i)) \quad (2)$$

Количество информации по всем вариантам является их суммой по всем вариантам выражения (2), которое при равновероятных вариантах сводится к выражению (1).

Для извлечения логарифма по основанию 2 можно воспользоваться выражением:

$$\log_2 (x) = \ln(x) / \ln(2)$$

## **ЛЕКЦИЯ №7. ВИЗУАЛЬНАЯ КУЛЬТУРА ДИЗАЙНА ИНТЕРФЕЙСА**

Занимаясь проектированием взаимодействия, мы получили описание интерфейса, которое содержит предельно ясную и точную информацию о том, кто конкретно будет использовать интерфейс, каким образом и с какой целью.

Однако, проектирование интерфейса не может концентрироваться только на поведении, необходимо учитывать внешний вид (форму) и информационное наполнение (содержание, контент) разрабатываемого интерфейса. Поэтому необходимо сочетать подходы различных дисциплин проектирования: проектирование взаимодействия, основное внимание на поведение; информационная архитектура, занимается структурированием содержания; графический дизайн, отвечает за внешний вид интерфейса и его функционала.

### **Визуальный дизайн интерфейсов**

**Дизайн** – это сознательные и интуитивные усилия по созданию значимого порядка.

К сожалению, огромная армия даже опытных, «модных» и эффективных дизайнеров забывают, что результатом их творчества должен быть интерфейс программного продукта или сайт, а не только «супер-скриншот» годный только для портфолио.

Художники и визуальные дизайнеры работают с одними и теми же изобразительными средствами, однако их деятельность служит различным целям.

Цель художника – создать объект, взгляд на который вызывает эстетический отклик. Изобразительное искусство – способ самовыражения художника. Художник не связан почти никакими ограничениями. Чем необычнее и своеобразнее продукт его усилий, тем выше он ценится. Дизайнеры создают объекты, которыми будут пользоваться другие люди.

Если говорить о дизайнерах визуальных интерфейсов, то они ищут наилучшее представление, доносящее информацию о поведении программы, в проектировании которой они принимают участие. Придерживаясь целеориентированного подхода, они должны стремиться представлять поведение и информацию в понятном и полезном виде, который поддерживает маркетинговые цели организации и эмоциональные цели персонажей. Разумеется, визуальный дизайн пользовательских

интерфейсов не исключает эстетических соображений, но такие соображения не должны выходить за рамки функционального каркаса.

Для успешного проектирования цифровых интерактивных продуктов, необходимо помнить о сложном поведении, которое эти продукты демонстрируют. Однако, силы, вложенные в разработку модели поведения программного продукта, будут потрачены впустую, если вы не сумеете должным образом донести до пользователей принципы этого поведения.

Визуальный дизайн интерфейсов – очень нужная и уникальная дисциплина. Она способна серьезно повлиять на эффективность и привлекательность продукта, но для полной реализации этого потенциала нужно не откладывать визуальный дизайн на потом, а сделать его одним из основных инструментов удовлетворения потребностей пользователей и бизнеса.

### **Графический дизайн и пользовательские интерфейсы**

Графические дизайнеры обычно очень хорошо разбираются в визуальных аспектах и хуже представляют себе понятия, лежащие в основе поведения программного продукта и взаимодействия с ним. Они способны создавать красивую и адекватную внешность интерфейсов, а кроме того – привносить фирменный стиль во внешний вид и поведение программного продукта. Для таких специалистов дизайн или проектирование интерфейса есть в первую очередь тон, стиль, композиция, которые являются атрибутами бренда, во вторую очередь – прозрачность и понятность информации и лишь затем – передача информации о поведении посредством ожидаемого назначения.

### **Визуальный информационный дизайн**

Информационные дизайнеры работают над визуализацией данных, содержимого и средств навигации.

Им необходимы некоторые навыки, которые присущи графическим дизайнерам, но они должны еще обладать глубоким пониманием и правильным восприятием роли поведения. Их усилия в значительной степени сосредоточены на организационных аспектах проектирования. В центре их внимания находится соответствие между визуальной структурой интерфейса с одной стороны и логической структурой пользовательской ментальной модели и поведения программы – с другой. Кроме того, их заботит вопрос о том, как сообщать пользова-



телю о состояниях программы и что делать с когнитивными аспектами пользовательского восприятия функций.

Усилия информационного дизайнера направлены на то, чтобы представить данные в форме, способствующей их верному истолкованию. Результат достигается через управление визуальной иерархией при помощи таких средств, как цвет, форма, расположение и масштаб. Распространенными объектами информационного дизайна являются всевозможные графики, диаграммы и прочие способы отображения количественной информации.

Чтобы создавать привлекательные и удобные пользовательские интерфейсы, дизайнер интерфейсов должен владеть базовыми визуальными навыками – пониманием цвета, типографики, формы и композиции – и знать, как их можно эффективно применять для передачи поведения и представления информации, для создания настроения и стимулирования физиологических реакций. Дизайнеру интерфейса также требуется глубокое понимание принципов взаимодействия и идиом интерфейса, определяющих поведение продукта.

### **Строительные блоки визуального дизайна интерфейсов**

Дизайн интерфейсов сводится к вопросу о том, как оформить и расположить визуальные элементы таким образом, чтобы внятно отразить поведение и представить информацию. Каждый элемент визуальной композиции имеет ряд свойств, и сочетание этих свойств придает элементу смысл. Пользователь получает возможность разобраться в интерфейсе благодаря различным способам приложения этих свойств к каждому из элементов интерфейса. В тех случаях, когда два объекта обладают общими свойствами, пользователь предположит, что эти объекты связаны или похожи. Когда пользователи видят, что свойства отличаются, они предполагают, что объекты не связаны.

Чтобы создать полезный и привлекательный пользовательский интерфейс, нужно учитывать визуальные свойства каждого элемента или группы элементов и тщательно поработать с каждым из этих свойств.

**Форма.** Какую форму имеет объект? Он круглый, квадратный или похож на амебу? Форма – главный признак сущности объекта для человека. Мы узнаем объекты по контурам. Силуэт ананаса (рис. 32), все равно позволяет нам понять, что это ананас.



Рисунок 32

При этом анализ форм требует большей концентрации внимания, чем анализ цвета или размера. Поэтому форма – не лучшее свойство для создания контраста, если требуется привлечь внимание пользователя.

**Размер.** Насколько велик или мал объект относительно других объектов на экране?

Более крупные элементы привлекают больше внимания, особенно если они значительно превосходят размерами окружающие элементы. Размер является переменной упорядоченной и поддающейся количественному определению, то есть люди, автоматически упорядочивают объекты по размеру и склонны оценивать их по размеру; если у нас есть текст

в разных размерах, предполагается, что относительная важность текста растет вместе с размером и что полужирный текст более важен, чем текст с нормальным начертанием.

Использование размера в информационной иерархии является эффективным способом привлечь внимание пользователя к конкретной части страницы. Потому что размер является одним из самых мощных форм организации визуальной композиции. Крупнейшие элементы должны быть самыми важными.

Таким образом, размер – полезное свойство для обозначения информационных иерархий. Достаточное расхождение в размерах обычно быстро привлекает внимание человека.

Размер – диссоциативное свойство. Это означает, что если объект очень мал или очень велик, становится сложно интерпретировать другие переменные, например форму.

**Яркость и контрастность.** Насколько темным или светлым является объект? Разумеется, понятия темного и светлого обретают смысл преимущественно в контексте яркости фона. На темном фоне темный текст почти не видно, тогда как на светлом он будет резко выделяться.

Значение яркости и контрастности может стать хорошим инструментом для привлечения внимания к тем элементам, которые требуется подчеркнуть. Оно также упорядоченная переменная – скажем, более темные (с более низкой яркостью) цвета на карте легко интерпретируются: они обозначают большие глубины или большую плотность населения.

Значение яркости тоже может быть диссоциативным. Если фотография слишком темная или слишком светлая, становится невозможно разобрать, что на ней.

**Контраст** при использовании должным образом может добавить драму, интригу и выдвинуть на первый план важные различия в функциональности. Всё сводится к примитивным человеческим инстинктам: мы предрасположены замечать различия, потому что это помогло нам выжить в какой-то момент эволюции.

Контраст показывает относительное значение. Изменения в размере текста и цвете фона говорит о том, что что-то изменилось, и требует внимания. Переход от светлого цвета фона к тёмному, помогает быстро отделить основное содержание страницы от колонтитула. Контраст между легким, воздушным разделом заголовка и темным, зеленым колонтитулом (рис. 33) показывает иерархию информации.



Рисунок 33

**Цвет.** Желтый, красный или оранжевый? Цветовые различия быстро привлекают внимание.

При этом в отличие от размера или яркости цвет изначально не обладает свойством упорядоченности и не выражается количественно, поэтому далеко не идеален для передачи информации такого рода. Кроме того, не следует делать цвет единственным способом передачи информации, поскольку цветовая слепота встречается довольно часто. По разным данным, той или иной формой цветовой слепоты страдают до 10%

мужчин, и использование, например, красного и зеленого цветов для указания контраста затрудняет работу с приложением для этих людей.

Использование контраста в цвете – жизненно необходимая техника, которую вам стоит взять на вооружение. Она помогает вести взгляд пользователя, менять его настроение, даёт подсказки.

Говоря о контрасте, цвета можно условно разделить на две противоположные группы: **тёплые и холодные**. Использование этих цветов обеспечивает контраст, и можно легко запомнить, какие цвета являются "теплыми", а какие цвета являются "холодными".

**Теплыми** являются те цвета, которые будут напоминать вам о лете, солнце или огне. Они составляют цвета от фиолетовых до желтых.

**Холодные** цвета могут напоминать весну, лед, или воду. Эти цвета простираются от желто-зеленого и до фиолетового. Эти цвета более склонны к расслаблению, успокоению, покорению и пассивности.

Например, окрашивание СТА-кнопки в красный может помочь привлечь к ней внимание, а размещение этой же кнопки на голубом или зелёном фоне сделает её ещё более заметной. Контраст между тёплыми и холодными цветами может обратить внимание (или отвлечь его) на определённые области в интерфейсе. Просто посмотрите на пример сайта TV Safety.org (рис. 34). Особенно хорошо видно, как красные навигационные кнопки выделяются, потому что они сопровождаются зелёным фоном.



Рисунок 34

На следующем примере (рис. 35) видно, как жёлтый ярко выделяется на синем. Таким же образом красный забирает внимание у зелёного, почти съедая его.

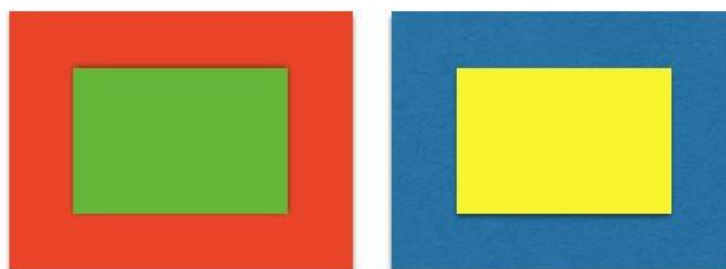


Рисунок 35

Применяйте цвет с умом. Чтобы создать эффективную визуальную систему, позволяющую пользователю выявлять сходства и различия объектов, используйте ограниченный набор цветов – эффект радуги перегружает восприятие пользователей и ограничивает возможности по передаче ему информации. Сайт Spectra (рис. 36) использует цвета для обозначения различных категорий новостей, так что легко найти определенный тип информации на основе ключа цвета.



Рисунок 36



## Цветовые модели

Цвета образуются в природе двумя противоположными путями. Во-первых, источники света (солнце, лампочки, экраны компьютеров и телевизоров) излучают свет различных длин волн, воспринимаемый глазом как цветной. Во-вторых, попадая на поверхность несветящихся предметов, свет частично поглощается, а частично отражается, и отраженное излучение воспринимается глазом как окраска предметов. Таким образом, цвет объекта возникает в результате излучения или отражения. Для описания излучаемого и отраженного цвета используются разные математические модели.

**Цветовая модель RGB.** Модель RGB описывает излучаемые цвета и основана на трех базовых цветах – **Red** (Красный), **Green** (Зеленый), **Blue** (Синий). Остальные цвета образуются при смешивании этих трех основных. При наложении одного компонента на другой яркость суммарного цвета увеличивается. При совмещении всех трех основных цветов получается серый цвет, который при большей яркости стремится к белому. В этой модели работают мониторы и бытовые телевизоры. Модель RGB называется аддитивной, а ее компоненты – красный, зеленый и синий – **основными цветами**.

**Цветовая модель CMYK.** Окрашенные несветящиеся объекты поглощают часть спектра белого света, падающего на них, и отражают оставшееся излучение. Чем больше краски находится на бумаге, тем больше света она поглощает и меньше отражает. Цвета, которые используют белый свет, вычитая из него определенные участки спектра, называются субтрактивными (вычитательными). Цветовыми компонентами этой модели являются не основные цвета, а цвета, которые получаются в результате вычитания основных цветов из белого: **Cyan** (голубой) = белый – красный, **Magenta** (пурпурный) = белый – зеленый, **Yellow** (желтый) = белый – синий. Эти три цвета являются дополнительными, потому что они дополняют основные цвета до белого.

Существенную трудность в полиграфии представляет получение черного цвета. Теоретически его можно добиться совмещением трех дополнительных или основных красок, но на практике результат оказывается негодным. Поэтому в цветовую модель CMYK добавлен еще один цвет – черный. Ему эта модель обязана буквой **K** в названии (**Cyan Magenta Yellow BlacK**).

**Цветовая модель HSB.** Модель HSB самая интуитивно понятная и удобная для человека. Она основана на цветах модели RGB, но имеет другую систему координат. В модели HSB тоже три компонента: оттенок



**Hue** (цвета), **Saturation** (насыщенность цвета) и **Brightness** (яркость цвета). Комбинируя эти три компонента, можно получить не меньшее количество произвольных цветов, чем в других цветовых моделях.

**Цветовая модель LAB.** Достоинством этой модели является ее независимость от способа производства цвета, в ее системе можно описывать как цвета печати, так и цвета, излучаемые монитором. Для построения модели LAB также используются три компонента: **Lightness** (Яркость) и **Chroma** (Интенсивность), которые вместе составляют информацию об **Luminance** (Освещенности) в изображении, содержащуюся в канале **L**. Канал **A** хранит информацию о тонах от зеленого до пурпурного, и, наконец, информация о тонах от голубого до желтого приходится на канал **B**.

### Цветовые схемы

Цвета в интерфейсах обычно подбираются исходя из цветовых схем. Цветовые схемы существуют уже очень давно. Хотя цвета Web отличаются от цветов печати, принципы остаются такими же. Одним из инструментов в сети, который позволяет быстро и легко определять цветовые схемы, и даже определить, обеспечивают ли выбранные цвета достаточную контрастность для слабовидящих или не различающих цвета пользователей, является Color Scheme Generator II (<http://www.colorschemedesigner.com/>).

Если требуется дополнительная помощь, чтобы определить, что выбранные цвета обеспечивают достаточно хороший контраст, попробуйте использовать инструмент Contrast Analyser (<http://www.paciellogroup.com/resources/contrast-analyser.html>) компании PacielloGroup. Этот инструмент определяет контраст между цветами фона и переднего плана.

Существуют следующие цветовые схемы:

**Монохроматические** – схема соответствует одному цвету и всем его оттенкам, тональностям и теням (рис. 37). Хотя эта схема самая простая для использования, она не вызывает большого энтузиазма при проектировании Web у многих дизайнеров.



Рисунок 37

**Дополнительные** – при выборе одного цвета и его противоположного, используют также все оттенки, тональности и тени обоих

цветов (рис. 38). Это обеспечивает более широкий диапазон выбора, и хорошо преобразуется с помощью сетевого инструмента цветов.



Рисунок 38

Сайт организации Гринпис (рис. 39), США (<http://www.greenpeace.org/usa/>) – хороший пример дополнительной цветовой схемы.



Рисунок 39

Он является одним из многих сайтов, которые используют контрастную цветовую схему. Вы видите желтые и оранжевые цвета, но доминирующими цветами являются зеленый и красный – два цвета, которые являются прямо противоположными друг другу на цветовом круге.

Если вы заметили, как цвета работают на круге, то скоро обнаружите, что не можете выбрать один цвет, не выбрав его противоположный по "температуре". Поэтому при выборе горячего красного цвета, противоположным будет холодный зеленый. Или, если вы вы-

Хорошим примером использования теплых/холодных цветов является сайт Ecolution (рис. 40).



Еvolution обычно использует красный в качестве акцентирующего цвета на своей домашней странице, как контраст своему зеленому логотипу. Затем они смешивают два контрастных цвета с различными оттенками, тональностями и тенями двух этих цветов. Даже черные цвета на изображении могут склоняться к «теплому» или «холодному», так же как и белые цвета. В целом, фотография будет «теплой», что хорошо сочетается с совершенно чистым зеленым цветом. Хотя они используют те же самые цвета, что и Greenpeace, сайт Eolution выглядит менее «слепящим», используя более насыщенные тональности и тени на фотографии. Всё не сложно, если вы знаете одно простое правило: тёплые цвета доминируют над холодными.

**Триадическая** цветовая схема (рис. 41) создается при выборе одного цвета и добавлении затем двух других цветов, которые должны лежать на одинаковом расстоянии друг от друга на цветовом круге, следующим образом:

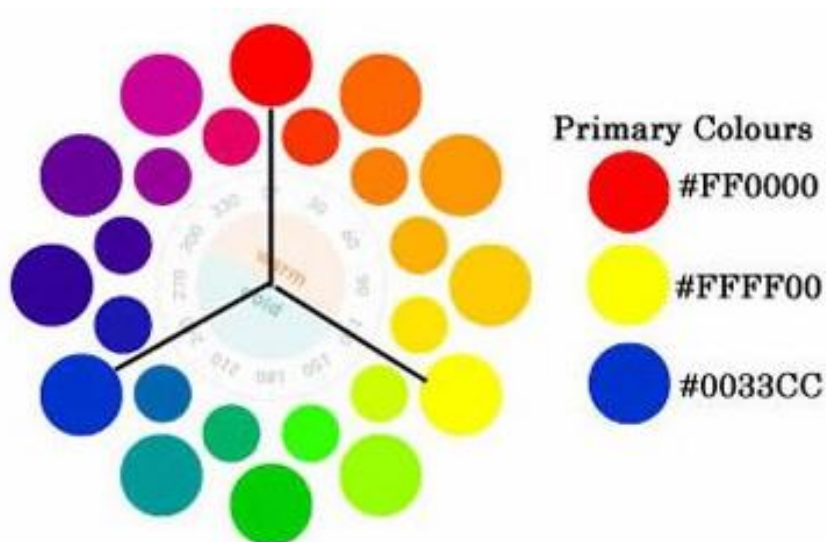


Рисунок 41

Триадическая цветовая схема содержит также теплые и холодные цвета, но одна температура будет преобладать. Обычно, температура, которая будет преобладать над другими, выбирается для переднего плана. На сайте Puzzle Pirates (рис. 42) (<http://www.puzzlepirates.com/>) – хорошая триадическая цветовая схема.



Рисунок 42

Они используют основную красно-сине-желтую схему, и эта основная схема отлично подходит для сайта детской игры. Синий цвет



является преобладающим, и что красные и желтые цвета используются как акценты и направляют движение глаз по странице.

**Тетрадические цветовые схемы.** Чем больше цветов выбирается, тем более сложной будет цветовая схема. Однако один из приемов состоит в выборе оттенка, тональности или тени и придерживаться этих областей повсеместно, а не смешивать чистые цвета и их оттенки, тональности и тени. Этот метод хорошо работает с четырехцветной тетрадической схемой (рис. 43).

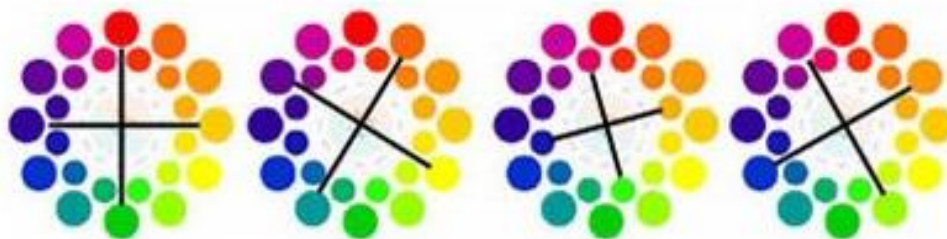


Рисунок 43

Тетрадическая цветовая схема – похожа на дополнительную схему, только используется две пары дополнительных цветов, расположенных на равном расстоянии друг от друга.

Сайт Jane Goodall Institute является (рис. 44) одним из немногих сайтов, которые действительно хорошо реализовали тетрадическую цветовую схему.



Рисунок 44

Обратите внимание на пурпурный цвет, желтую тональность, красные отблески на фотографии (сайт содержит еще красный цвет далее вниз на странице), и зеленые цвета.

Трендом 2016 года станут насыщенные цвета и неоновые палитры. Пример – сайт Bloomberg (рис. 45).

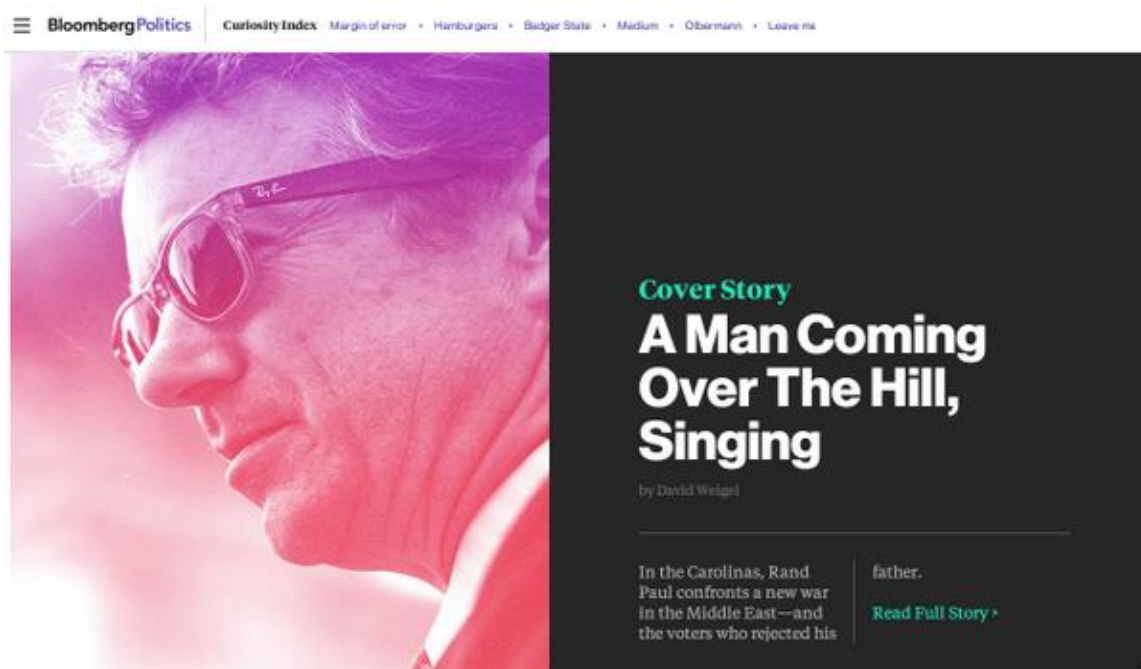


Рисунок 45

## Цветовая гармония

**Цветовая гармония** - это сочетание отдельных цветов или цветowych множеств, образующие органическое целое и вызывающие эстетическое переживание.

Цветовая гармония в дизайне представляет собой определенное сочетание цветов с учетом всех их основных характеристик, таких как: цветового тона, светлоты, насыщенности, формы, размеров занимаемых этими цветами на плоскости, их взаимного расположения в пространстве, которое приводит к цветовому единству и наиболее благоприятно эстетически воздействует на человека.

Признаки гармонии цвета:

**Связь и сглаженность.** Связующими фактором может быть: монохромность, ахроматичность, объединяющие подмеси или налеты (подмесь белого, серого, черного), сдвиг к какому-либо цветовому тону, гамма.



**Единство противоположностей, или контраст.** Не забывайте о контрастах. Виды контраста: по яркости (темное-светлое, черное-белое и т.д.), по насыщенности (чистые и смешанные), по цветовому тону (дополнительные или контрастные сочетания).

**Мера.** Не используйте без необходимости слишком яркие цвета в оформлении сайта. Сайт не должен быть слишком «пестрым».

**Пропорциональность, или соотношение частей** (предметов или явлений) между собой и целом. В цветовой гамме – это подобие отношений яркостей, насыщенности и цветовых тонов.

**Равновесие.** Цвета в композиции должны быть уравновешены.

**Ясность и легкость восприятия.** Используйте цветовую гамму, наиболее приятную для глаз.

**Прекрасное, стремление к красоте.** Недопустимы психологически негативные цвета, диссонансы.

### **Шрифтовая схема**

Шрифты, называют также «гарнитурами шрифтов» и используют для вывода текста, чисел, букв, и других символов.

Многие исследования показывают, что множество гарнитур шрифта на Web-сайте может путать читателя. С другой стороны, сайт, который использует везде только один шрифт, кажется безвкусным.

Потому при построении дизайна лучше использовать гарнитуру одного-трех шрифтов: **базовый** шрифт – основной шрифт материалов сайта и **акцидентный** – шрифт для заголовков.

В некоторых случаях вводятся дополнительные шрифты для: меню и навигации, блоков выделения (важной информации, цитат, выносок), для мелкого текста, с целью повысить читабельность.

Дизайнер должен спланировать единую общую схему размеров отступов/заступов для всех элементов на сайте, иерархию заголовков и навигационных элементов (например, для древовидного меню или облака тегов). Она должна быть цельной и использоваться на всех страницах сайта. Всё последующее оформление информации на сайте должно строиться на базе общей схемы.

На рисунке 46 представлена шрифтовая схема простого корпоративного сайта.



Рисунок 46

Существует четыре следующие основные типы шрифтов:

**Шрифт с засечками (serif).** Любая гарнитура шрифта, которая содержит завершающие штрихи, расширяющиеся или сужающиеся концы, или имеет реально отсеченные окончания (включая прямоугольные засечки). В ходе истории шрифт с засечками был выбран для печати основного текста, так как его легко читать на печатной странице. Но интерфейс отличается от печати, и исследования показывают, что шрифты без засечек по удобочитаемости шрифта, читать легче, чем с засечками. Пример типа шрифта с засечками, Times New Roman, нормального (не жирного и не курсива), размером 18 pt (пунктов) на рисунке 47.

Times New Roman, regular, 18 point

Рисунок 47

**Без засечек (sans-serif).** Любая гарнитура шрифта, конечные штрихи которой не имеют никаких расширений, пересекающих штрихов, или других украшений. Его используют для основного текста для Web. Пример шрифта без засечек, Verdana, нормального, размером 18 pt на рисунке 48.

## Verdana, regular, 18 point

Рисунок 48

**Рукописный или курсив.** Эти шрифты обычно выглядят по большей части как написанные пером или кистью. Эти шрифты будут включать те, которые кажутся рукописными, даже хотя и не являются курсивом. Одной из причин отказа от использования этого типа шрифта на Web-странице, особенно в основном тексте, является трудность чтения в больших отрывках (подумайте о том, как трудно было бы читать написанное от руки письмо, или манускрипт двенадцатого века, который можно увидеть в музее).

Кроме того, не все браузеры выводят один и тот же шрифт, поэтому если вы решите использовать рукописный и курсивный шрифт, то он может быть выведен как шрифт с засечками в каком-то другом браузере. Пример рукописного шрифта *Staccato*, размером 18 pt на рисунке 49.

*Staccato222BT, regular, 18 point*

Рисунок 49

**Специальные шрифты,** включая моноширинный. Единственным критерием моноширинного шрифта является единая фиксированная ширина всех символов, аналогично тому.

Некоторые шрифты могут иметь **фантастический внешний вид** эти шрифты используются единственно с декоративной целью. Эти шрифты могут найти применение в небольшой области, такой как заголовки или в рекламных объявлениях. Пример специального шрифта, Jokewood, размером 18 pt на рисунке 50.

**JOKEWOOD, REGULAR, 18 POINT**

Рисунок 50

Моноширинные шрифты нашли свое применение на Web-сайте, особенно при выводе программного кода (<http://www.lowling.org/fonts/>) (рис. 51).

- TrueType
- Clear, Dark, slashedzeros.
- AvailablewithfreePovraysoftwar
- 

Рисунок 51

Существуют также и другие различия.

- Не все типы шрифта создаются одинаковыми, даже если создаются одного размера в пунктах. Размер в пунктах определяет высоту букв, и некоторые шрифты будут больше при 18 pt, чем другие.
- Расстояние между буквами и словами могут быть разные, или некоторые гарнитуры шрифтов, такие как Jokewood, не имеют букв нижнего регистра.
- Шрифты могут выглядеть по-разному во всех браузерах, так как различные браузеры остаются по сути несовместимыми. Причина этой проблемы состоит в том, что не все операционные системы поддерживают одни и те же шрифты. И даже если одни и те же шрифты, то вариант, толщина и другие факторы могут представляться по-разному в том или ином браузере.

Тип шрифта является одним из объектов, над которым вы имеете некоторый контроль, но только если вы сохраняете его как можно более простым. В Web-приложениях обычно используют в качестве основного шрифта – Verdana, а для заголовков – Times Roman или Georgia.

Даже на небольшом экране у пользователя не должно быть проблем с чтением текста. Если оптимальный размер шрифта для отображения на большом экране составляет 14 pt, то для мобильного устройства он должен быть как минимум в два раза больше. Следует, однако, учитывать, что чем крупнее шрифт — тем меньше информации удастся разместить на сайте.

### **Композиция: вид и ощущение приложения**

Все элементы дизайна – от типографики до иконографии – относятся к общей композиции. Каждый экранный интерфейс или веб-страница вызывают у посетителя внутренние эмоции еще до того, как он успевает заметить мелкие детали. Именно поэтому, разрабатывая дизайн, необходимо уделять особенное внимание композиционным методам проектирования.

Элементарные принципы дизайна композиции, цвета, и т.д. точно так же применяются к компьютерному экрану, как к листу бумаги или холсту. Композиция или размещение вашей страницы не только влияет на ее эстетическую привлекательность, но также имеет огромное воздействие на применимость вашего приложения.

Как сказал кто-то из великих, создать гениальную картину – это положить в нужные места нужные краски. То же самое, и с дизайном интерфейсов.

Композиция, если не вдаваться в сложности – это определенное сочетание частей изображения (соотношение их размеров, пропорции, цветов, фактур и т.д.).

### Правила композиции

**Целостность** – в правильной композиции ни один из элементов нельзя изъять, добавить или передвинуть без ущерба для целого.

Для нахождения целостной композиции обычно рассматривают будущее изображение как набор пятен – силуэтов отдельных элементов, которые komponуют на плоскости до достижения необходимого эффекта. Все элементы композиции должны быть связаны чем-либо воедино – стилем рисунка, выравниванием, цветами, размерами и т.д. (пример на рис. 52). Применительно к веб-дизайну – не может быть целостной композиция, где дизайн страницы никак не перекликается с логотипом.



Рисунок 52

**Выразительность** – неформализуемое качество композиции, проявляется в том, что изображение быстро захватывает внимание зрителя, ясно показывает процессы, которые дизайнер хотел отобразить. Фактически – это соответствие вашей идеи и формы, найденной для ее выражения. Когда зрители не понимают вашу идею – скорее всего, хромает композиция, а не восприятие зрителя.

Выразительность проявляется в умелом использовании контрастов по цветам, светлоте, размерам. Контраст тем выразительней, чем больше он соответствует идее изображения (напр. контраст заголовка и текста по размеру на рис. 52 соответствует идее большей важности заголовка).



Рисунок 53

**Выявление центра** – подчеркивание той части, которая выражает главную идею.

Мы строим композицию на плоскости. Будь то фотография, лист бумаги или монитор компьютера. Если через эту плоскость провести две диагональные линии (рис. 54), точка их пересечения укажет на **геометрический центр** нашей будущей композиции.



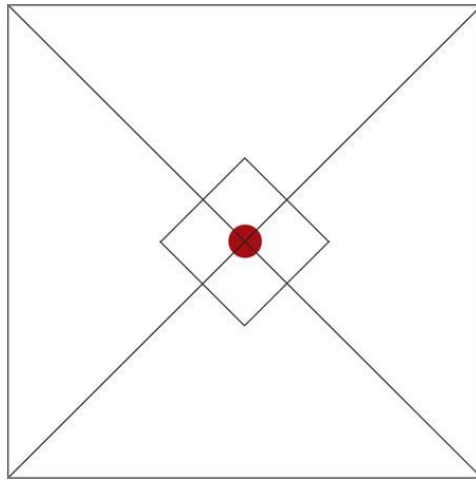


Рисунок 54

Любой предмет, вписанный в этот центр, будет чувствовать себя вполне уверенно.

**Композиционный центр** служит для фокусировки внимания зрителя на деталях композиции. Как правило, центр находится где-то чуть выше середины экрана, но он может быть смещен каким-либо "активным" элементом композиции – ярким, большим предметом, другим контрастным объектом.

Композиционный центр и геометрический центр композиции могут не совпадать. Композиционных центров в композиции может быть несколько, в то время, как геометрический центр один. Композиционный центр может быть выделен: контрастом света и тени, контрастом цвета, размером, формой.

На практике часто используют метод выделения центра путем вписывания композиции в простую геометрическую фигуру – квадрат, ромб, треугольник, круг, овал. Типичный пример – использование линий, соединяющих логотип и кнопки либо очерчивающих весь экран и захватывающих кнопки.

**Гармония** – это слаженность. Единое целое, в котором все элементы дополняет друг друга. Некий единый механизм.

Самый большой такой механизм – это окружающий нас мир, в котором все элементы взаимосвязаны. Нет ничего более гармоничного, чем сама природа. Поэтому и понимание гармонии приходит к нам от нее. А в природе огромное количество зрительных образов подчиняется двум правилам: **симметрии** и правилу «**золотого сечения**».

**Симметрия** – полезное средство организации интерфейса с точки зрения достижения визуального равновесия. Несимметричные интерфейсы обычно выглядят так, словно вот-вот завалятся на один бок.

Опытные дизайнеры способны достигать асимметричного равновесия, управляя визуальным весом отдельных элементов. Тест «с прищуриванием» позволяет проверить сбалансированность интерфейса.

В интерфейсах чаще всего применяют два типа симметрии: *вертикальная осевая симметрия* (симметрия относительно вертикальной линии, проведенной через центр группы элементов) и *диагональная осевая симметрия* (симметрия относительно диагонали). В большинстве приложений присутствует симметрия одного из этих типов.

Симметрия встречается во всех типах дизайна. Встречается и в интернете. Вот некоторые примеры симметрии в дизайне веб-сайта.

Базовая схема веб-сайта Apple Store (рис. 55) устанавливается в симметричную сетку. Если сложить макет сайта Apple Store по середине, обе стороны будут совпадать. Уберите содержание колонок, и вы увидите блоки которые составляют генеральный план макета (рис. 56).

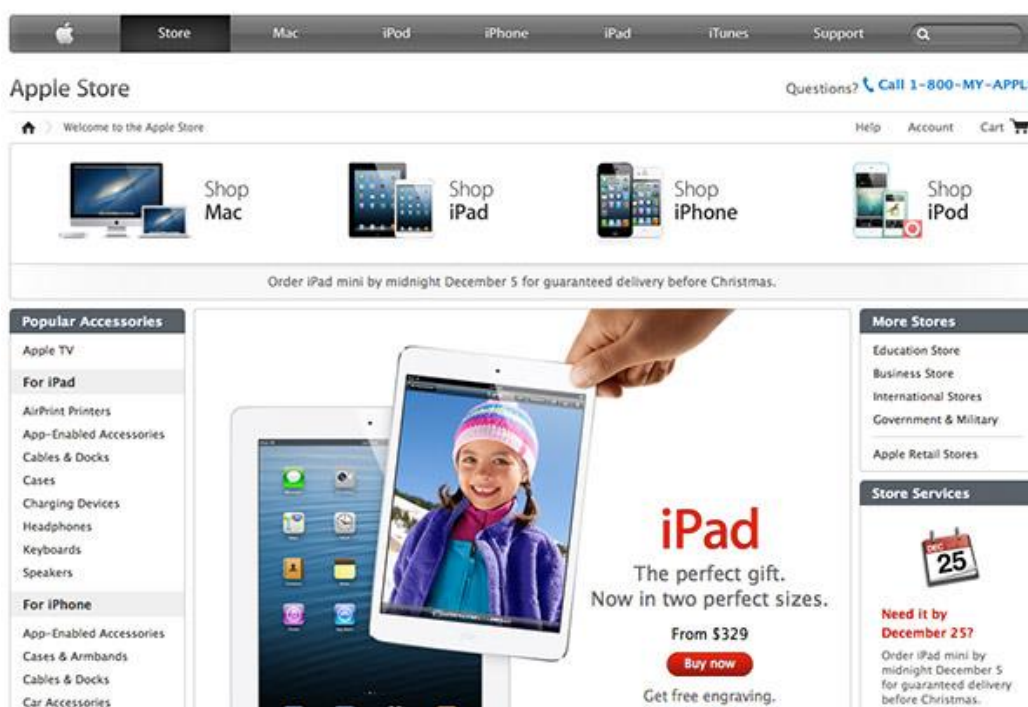


Рисунок 55



Рисунок 56

Вот еще один хороший пример симметрии в дизайне сайта. Обратите внимание, что изображения на ноутбуке (рис. 57) и планшете будут асимметричными, но в целом, если разбить макет на основные блоки, они будут симметричными. Чуть нарушение симметрии с изображениями создает более интересную планировку, чем если бы оба изображения были бы пропорционально.

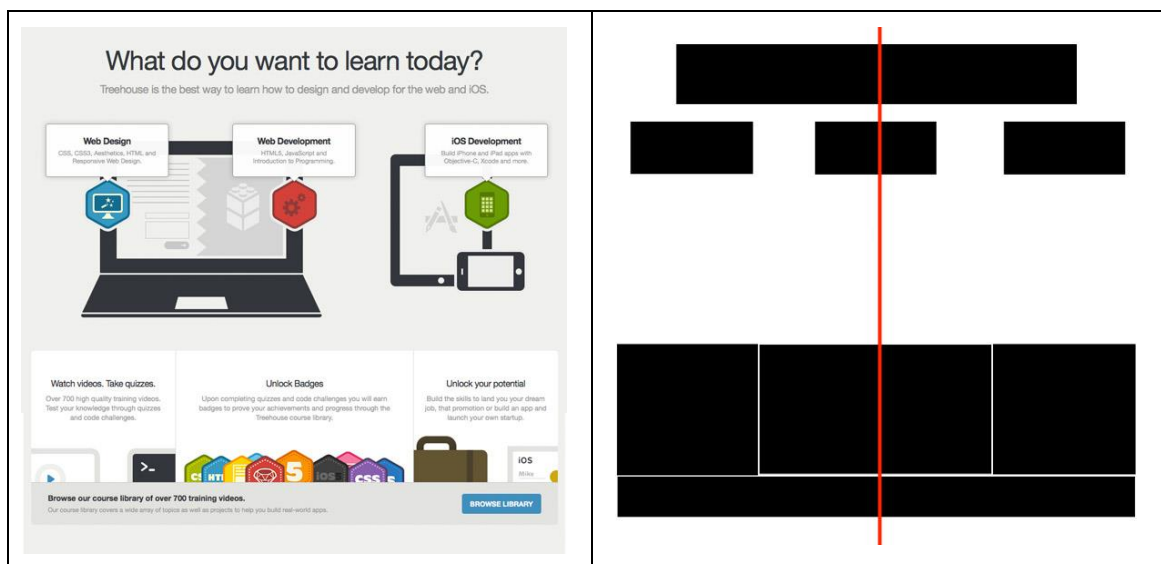


Рисунок 57

Дисбаланс композиции будет, как правило, тогда эффектно смотреться, когда он сделан осознанно, а не допущен по незнанию.

**Асимметрия** – отсутствие симметрии не означает отсутствие гармонии и баланса. Асимметричный дизайн может создать ощущение энергии или напряженности (рис. 58), но чувство композиционной гармонии зависит от ее использования.

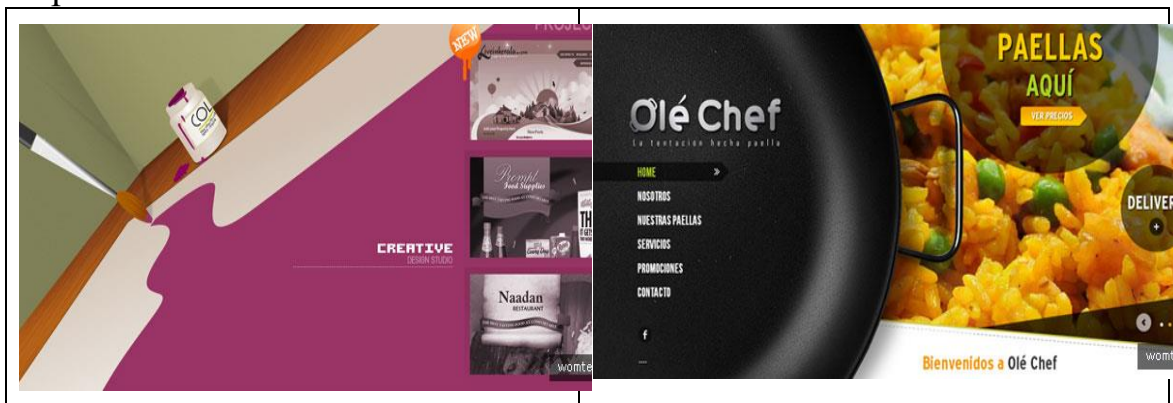


Рисунок 58

Чаще всего асимметричные методы проектирования используются в сочетании с элементами, которые имеют симметрию. Фокус в том, как именно вы сочетаете эти объекты. Это больше, чем просто картинка и пара слов.

Различное расположение элементов на плоскости может создать гармоничное или негармоничное изображение. Гармония – это ощущение и понятие о правильном расположении элементов весьма интуитивно. Однако можно выделить несколько совсем не интуитивных правил: **пропорции** (соотношения размеров) и **взаимное расположение элементов**.

**Пропорции.** Соотношение размеров либо разных объектов, либо составных частей или разных измерений (например, ширины и высоты) одного объекта – называют в дизайне **пропорцией**.

Архитекторы и художники древности придавали огромное значение выбору пропорций. Некоторые из их открытий, такие как «золотое сечение», вполне могут пригодиться и сегодняшнему дизайнеру, но в целом классический подход к определению пропорций, обычно заключающийся в вычислении размеров по формулам, в современном дизайне применяется редко.

«**Золотое сечение**» – также известное, как «Божественная пропорция», «золотая спираль» (рис. 59) и т.п. Это соотношение есть во всех аспектах жизни, от строения костей человека до спирального

расположения семян подсолнуха и завитков раковин моллюсков, оно лежит в основе всех биологических структур и кажется геометрической схемой самой жизни.

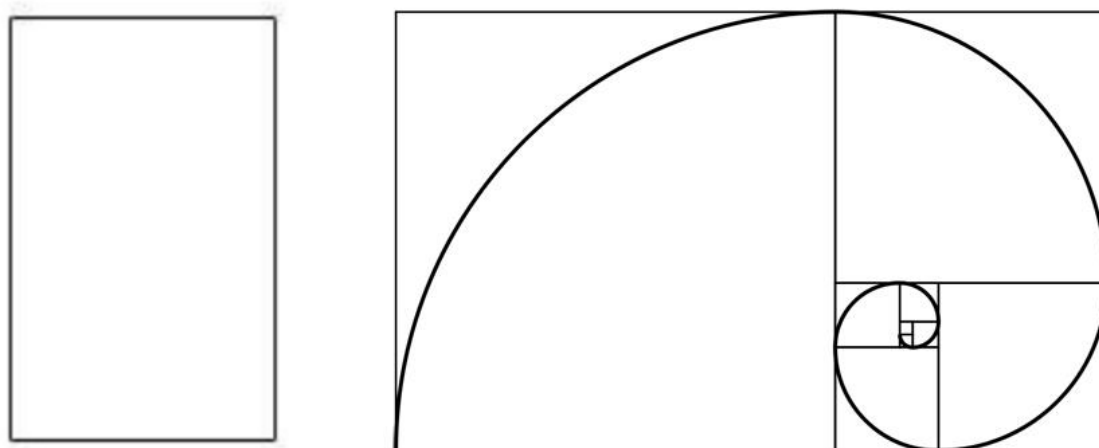


Рисунок 59

«Золотое сечение» можно получить, если разделить отрезок на две неравные части (рис. 60) таким образом, чтобы отношение всего отрезка (в нашем случае **С**) к большей части (**В**) равнялось отношению большей части отрезка (**В**) к меньшей (**А**).

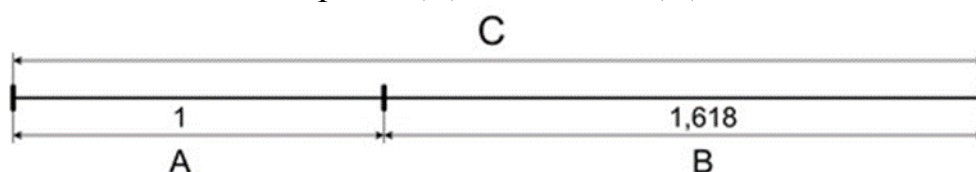


Рисунок 60

В пропорциях это будет выглядеть следующим образом.

$$\frac{C}{B} = \frac{B}{A} \quad \text{или} \quad \frac{\sqrt{5}-1}{2} = 0,618 \quad \text{и наоборот} \quad \frac{\sqrt{5}+1}{2} = 1,618$$

В процентном соотношении – это деление какой-либо величины в отношении 62 % и 38 %.

Части этого отрезка примерно равны **5/8** и **3/8** от всего отрезка. То есть, по правилу «золотого сечения» зрительные центры в изображении будут располагаться так (рис. 61):

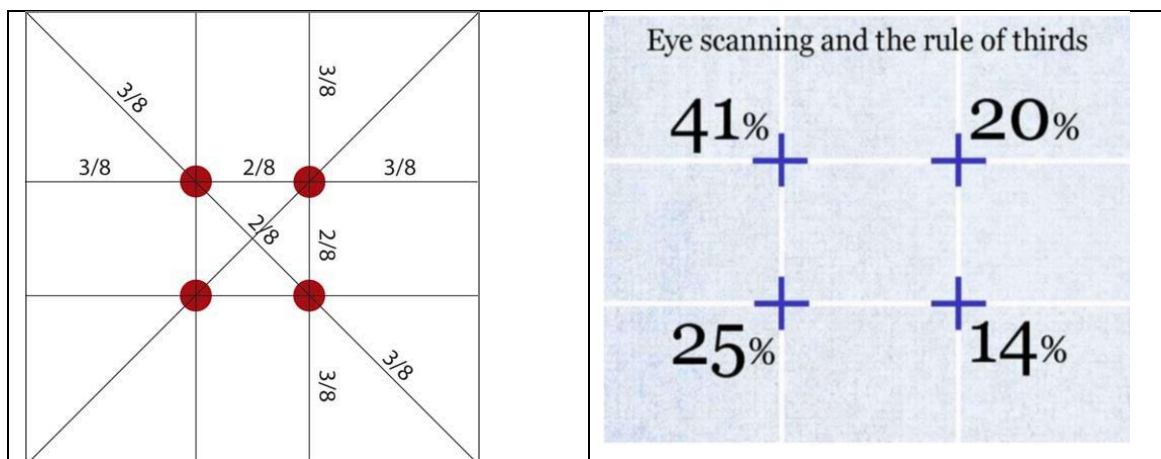


Рисунок 61

Любая из четырех точек пересечения линий – это превосходное место для расположения фокуса. Правило трех частей композиции определяет четыре идеальных места для фокальных точек. Новый дизайн Твиттера (рис. 62) был запущен в конце 2011 года, при этом применялось правило «золотого сечения» для построения макета сайта. Боковая лента и панель новостей вместе отражают метод «золотого прямоугольника». По словам Дуга Баумена, креативного директора сайта, там была преднамеренно использована магическая формула.

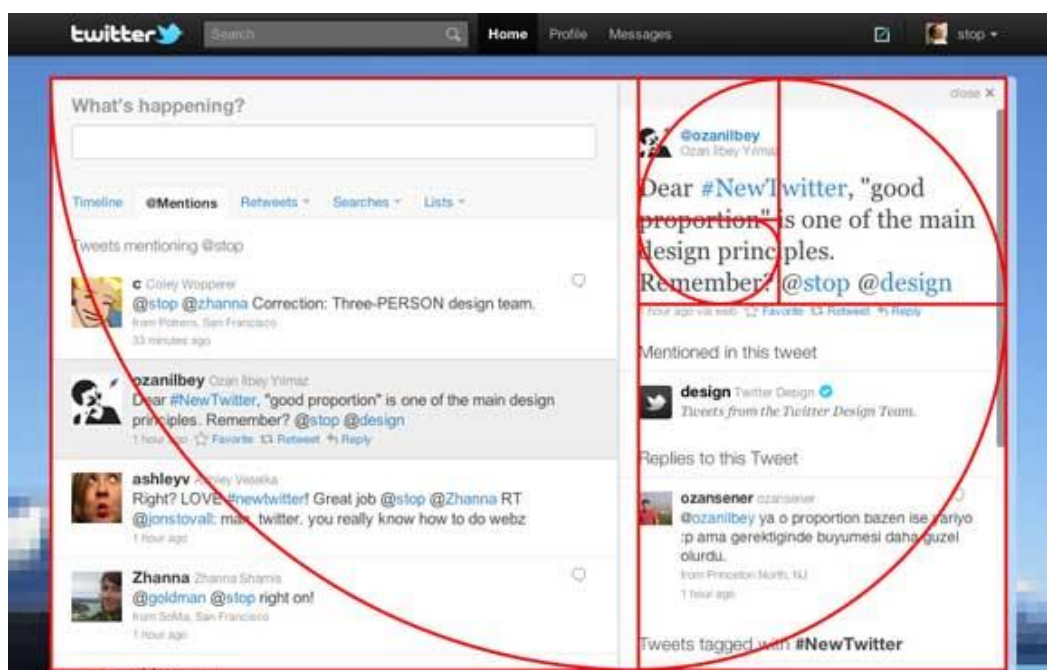


Рисунок 62



Говорят, что, многократно повторив «золотую пропорцию» в дизайне, можно достичь невиданного успеха, но все-таки не воспринимайте эти догмы слишком близко к сердцу: во-первых, из каждого правила есть исключения и, во-вторых, помимо «золотого сечения», есть и другие, не менее важные правила. Например, величина, обратная квадратному корню из двух (примерно 1:1,41), которая является основой международного стандарта размера бумаги (например, листа А4). Субъективность размеров, о которой мы говорили, делает субъективными и пропорции.

Современный дизайнер выбирает пропорции почти исключительно «на глазок», добиваясь нужного ему соотношения активностей элементов не только варьированием размеров, но и с помощью множества других инструментов.

### **Размеры экрана**

Дизайн интерфейса сайта и приложения, в отличие от простого графического дизайна, должен следовать не только общим художественным требованиям и последним трендам, но также быть в достаточной степени понятным и универсальным. Уровень технического развития на современном этапе позволяет выходить в сеть интернет даже с телевизора и некоторых моделей холодильников. Это не шутка. Адаптивность дизайна пользовательского интерфейса заключается в универсальном визуальном отображении и адекватном расположении элементов интерфейса на большинстве устройств с доступом в сеть интернет. В наше время, когда у всех пользователей самые различные мониторы, есть смысл разобраться в том, как будет отображаться та или иная информация на мониторах разного разрешения.

Существуют четыре типа экранной компоновки: фиксированной ширины, жидкая (или «резиновая»), эластичная и гибридная. Каждый тип имеет свои достоинства, ограничения и неоднозначности.

**Фиксированная компоновка.** При компоновке этого типа ширина ограничена определенным количеством пикселей (px) – в настоящее время чаще всего 960 px. Это число лучше всего подходит для макетов на основе модульных сеток, т. к. оно делится на 3, 4, 5, 6, 8, 10, 12 и 15, а значит, обеспечивает разные варианты сеток. Компоновка с фиксированной шириной чаще всего реализуется в сети, т. к. она позволяет создавать графически насыщенные варианты дизайна, имеющий одинаковый вид на различных экранах.

Однако в современной насыщенной экосистеме устройств, проблемы будут и у посетителей со слишком большими мониторами (у

них справа появится незапланированное белое пространство) и у владельцев смартфонов и планшетов, которые уменьшают масштаб таких сайтов, чтобы поместить их на экране.

**Резиновая компоновка.** В этом варианте компоновки размеры задаются не в пикселах, а в процентах, что значительно увеличивает приспособляемость конечного результата. В результате не важно, какое устройство использует пользователь.

**Эластичная компоновка.** При этой компоновке ограничения определяются размером шрифта и измеряются в единицах  $em$ .  $1em$  равен высоте используемого по умолчанию шрифта. Предположим, размер основного текста 16 px. В этом случае  $1em$  равен 16 px, а  $2em$  – 32 px.

К сожалению, эластичная компоновка не гарантирует отсутствия горизонтальной прокрутки. Если при размере шрифта 16 px вы зададите ширину контейнера 55  $em$ , на любом экране с разрешением ниже 880 px ( $16 \times 55$ ) появится горизонтальная прокрутка.

**Гибридная компоновка** – представляет собой комбинацию двух или трех ранее предложенных вариантов. Допустим, что под объявление выделено пространство в 300 px. Можно сделать боковую панель фиксированной ширины 300 px, а остальные столбцы можно задать в процентах. Это дает возможность поддерживать разработанную для объявления графику, остальная часть компоновки будет растягиваться.

Каждый из подходов имеет свои сильные стороны и свои ограничения. В конечном итоге все зависит от конкретного проекта. Однако наиболее ориентированы на будущее варианты гибкой компоновки.

Итак, делая «резиновый» дизайн не забываем, что:

- Общая композиция не должна нарушаться ни при каком разрешении монитора у пользователя.
- Все элементы масштабируются в зависимости от размера экрана пользователя и размера шрифта.
- Вся модульная сетка, блоки и прочие горизонталы масштабируются в процентах.
- Все шрифты, отступы, почти все вертикали масштабируются в  $em$  ( $1em$  – текущий размер шрифта). Можно брать любые пропорции от текущего шрифта:  $2em$ ,  $0.5em$  и т.п. Во многих случаях, это касается даже рамок (*border*).
- Исключение могут составлять лишь картинки. И то, жесткий размер в px для многих изображений – ограничение лишь по вертикали.

## Сетка

**Сетка** – один из самых мощных инструментов визуального дизайнера. После того как проектировщики взаимодействия определили общую инфраструктуру приложения и элементов его пользовательского интерфейса, дизайнеры интерфейса должны организовать композицию в структуру в виде сетки, которая будет должным образом подчеркивать важные элементы структуры и оставлять жизненное пространство для менее важных элементов и элементов более низкого уровня.

Модульные сетки предназначены для «устаканивания» содержания. Они должны передавать общую идею дизайна и взаимосвязи элементов, основанных на пространственной иерархии и позиционировании. Не существует одной единственной мастер-сетки, но есть общие принципы, которым необходимо следовать.

Преимущества работы с модульной сеткой в дизайне:

- Сетка позволяет представить информацию на странице цельной, упорядоченной и гармоничной.
- Сетка позволяет ускорить процесс работы, снизить временные затраты на поиски свободного или подходящего места в макете.
- Сетка позволяет добавить новую информацию, при которой не нарушится общая концепция дизайна.
- Разработанная модульная сетка может стать основой для построения дизайна для других страниц на сайте.
- Сетки позволяют пользователю облегчить восприятие материала, и помогают найти нужную информацию, при переходе по страницам.

При работе с построением макета дизайна необходимо использовать различные изображения, блоки текста, рекламы и других элементов так, чтобы они подходили друг другу и смотрелись как единое целое. Правильным решением будет использование одной структуры сетки для всего шаблона сайта

Сетка обеспечивает однородность и последовательность структуры композиции, единый каркас и схему расположения всех основных блоков и элементов.

В самом простом понимании это решетка из ячеек, где одна из них взята за основную единицу измерения (модуль), а остальные равны или кратны ей. В модульной сетке нет места случайности, все должно быть математически точно.

Сетки бывают простые и сложные, гибкие в использовании и не очень. Это не столь важно. Важно то, что если вы при проектировании

дизайна задали определенную сетку модулей – будьте добры, ей следовать от первой до последней страницы своего проекта. А если в процессе прорисовки внутренних страниц у вас появились элементы, не лежащие в принятую сетку – значит, вы недостаточно времени уделили ее проектированию.

Следование единой модульной сетке в рамках проекта не только увеличит цельность и логичность восприятия сайта, но и многократно упростит труд разработчикам.

Рассмотрим два макеты страниц представленных на рисунке 63.



Рисунок 63

Хотя оба эти изображения – всего лишь несколько прямоугольников, набор в верхней части кажется принципиально лучше, чем в нижней. Мы можем мгновенно распознать модель, принять её и двигаться дальше. Изображение внизу же вызывает визуальный дискомфорт, так как не имеет четкой картины, порядка, или цели и выглядит как случайный набор форм.

Выравнивание визуальных элементов – один из главных приемов, позволяющих дизайнеру представить продукт пользователям в систематизированном и упорядоченном виде.

Сетки позволяют стабилизировать структуру интерфейса и предоставляют дизайнеру логический шаблон для его создания.

Как правило, сетка делит экран на несколько крупных горизонтальных и вертикальных областей. Качественно спроектированная сетка задействует понятие шага, то есть минимального расстояния между элементами. К примеру, если шаг сетки составляет четыре пик-

села, все расстояния между элементами и группами должны быть кратны четырем.

В идеальном случае сетка должна задавать и пропорции различных областей экрана. Такие отношения обычно выражаются дробями. Среди распространенных дробей – прославленное «золотое сечение», число, равное примерно **1:1,618**.

Использование сетки в визуальном дизайне интерфейсов дает ряд преимуществ:

- **Удобство применения.** Поскольку сетка делает расположение элементов единообразным, пользователи быстро приобретают навыки поиска нужных элементов в интерфейсе. Последовательность в расположении элементов и выборе расстояний между ними облегчает работу механизмов визуальной обработки в мозгу человека. Качественно спроектированная сетка упрощает восприятие экрана.

- **Эстетическая привлекательность.** Аккуратно применяя сетку и выбирая подходящие соотношения между различными областями экрана, дизайнер может создать ощущение порядка, который удобен пользователям и стимулирует их работу с продуктом.

- **Эффективность.** Создание сетки и включение ее в процесс на ранних этапах детализации проектных решений сокращает число итераций и действий по "доводке" интерфейса. Качественная и явно обозначенная сетка закладывает основу для легко модифицируемого и расширяемого дизайна, позволяя разработчикам находить хорошие композиционные решения.

## Виды сеток

Самый простой вид сетки – **блочная сетка** (рис. 64). В западной литературе ее также называют «manuscript grid». Представляет собой грубо размеченную область – блок.

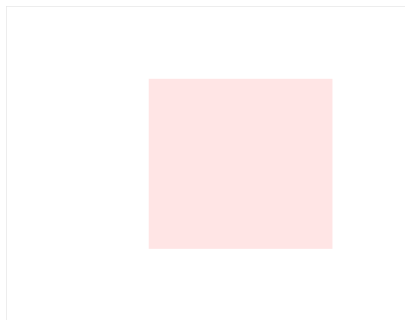


Рисунок 64

**Колоночная сетка** – состоит только из вертикального членения на колонки (рис. 65).

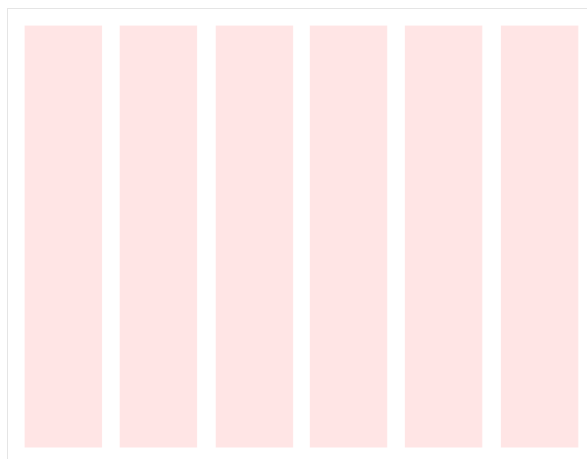


Рисунок 65

Объекты располагаются опираясь на эту сетку. Какой-то блок может занять у вас две колонки, какой-то – четыре. Поскольку используется единый модуль, и вы сохраняете отступы одинаковыми, дизайн всегда выглядит целостно и логично.

**Модульная сетка** (рис. 66) характеризуется наличием, как вертикального членения, так и горизонтального. То, что образуется на пересечениях, есть модуль.

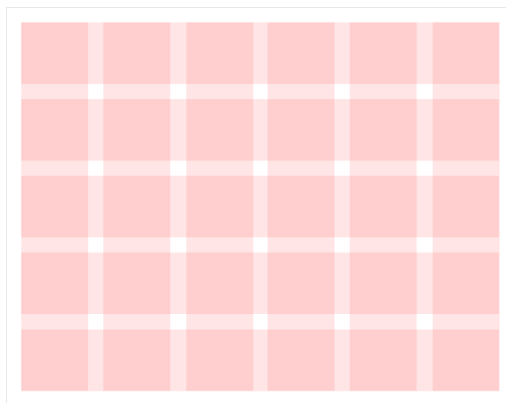


Рисунок 66

**Иерархическая сетка** – состоит из блоков размещенных интуитивно и не поддающихся закономерностям (рис. 67).



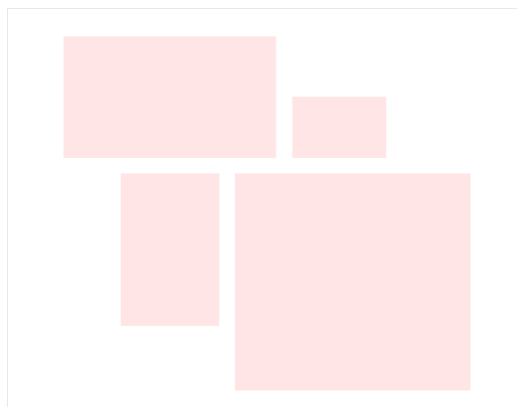


Рисунок 67

Сетки могут быть как простыми – с одинаковыми по размерам модулями, так и сложными, с нелинейными пропорциями у размеров модулей (рис. 68). К сложным пропорциям относятся:

- **«Золотое сечение».**
- **Ряд Фибоначчи** (каждое последующее число оказывалось равным сумме двух предыдущих: 1, 2, 3, 5, 8, 13, 21 и т. д).
- **«Предпочтительные числа»** (ряд чисел геометрической прогрессии, где каждое последующее число образуется умножением предыдущего числа на какую-нибудь постоянную величину).



Рисунок 68

Разумеется, это не все популярные пропорции.

**Шрифтовая сетка** обеспечивает вертикальный ритм. Необходимо выбрать высоту строки единую для всего макета. Все элементы рубрики с кеглем, отличным от кегля основного текста, должны иметь

междустрочный пробел кратный выбранной высоте строки. Высота каждого такого элемента (в сумме со всеми вертикальными полями) должна содержать целое количество строк шрифтовой сетки.

Таким образом, мы получаем прообраз будущей сетки – «зебру». На этой сетке будет лежать весь текст: абзацы, списки, заголовки, иллюстрации, плашки и прочее.

Чаще всего используются такие параметры для шрифтовой сетки: кегль — **12 пикселей**, высота строки — **18 пикселей**. — 12 пикселей, высота строки — 18 пикселей.

Стоит отметить, что существует два способа создания сетки шаблона: создать свою собственную сетку и скачать готовый шаблон сетки.

В интернете есть много готовых шаблонов сеток, которые вы можете использовать при создании интерфейса. Они помогают существенно сэкономить время. Это:

- **960.gs** – пожалуй, самый популярный инструмент для создания сеток.

- Сеточные шаблоны для Photoshop

<http://www.ravelrumba.com/photoshop-grids/>

- Плагины для создания сеток в Photoshop: GuideGuide;

- Модульная сетка (<http://modulargrid.org/#panel>);

- GridMaker2 (<http://andrewingram.net/2007/may/gridmaker-reboot/>);

- Скрипт для создания сетки из шейпов

(<http://www.agasyanc.ru/modgrid>).

### **Как разместить элементы управления**

Разобравшись в модульных сетках, можно начинать детализировать работу. В намеченную сетку нужно вписать все те элементы (или просто застолбить для них место), о которых написано в техническом задании.

Кто-то начинает с поиска места для логотипа. Кому-то больше нравится сначала найти подходящее место для графических элементов, а затем вокруг этих блоков располагать все остальное. Это в чем-то напоминает танцы «от печки»: когда вы сначала ставите в макет один элемент, потом другой, третий ...

Как же располагать информацию на странице? Есть много правил на этот счет, которые в основном образовались исторически, просто потому что все так делают. Давайте познакомимся с ними.

**Логотип.** Логотип должен располагаться слева и только слева, – так говорят консерваторы и никогда не отступают от этого правила. А вот и нет. Логотипу весьма желательно находиться вверху страницы, на первом экране, чтобы он был виден сразу же после загрузки стра-

ницы и не требовал вертикальной прокрутки (это уже гораздо ближе к правде). Где вы его разместите – сугубо ваше дело. Просто логотип слева – это традиция, которую редко нарушают, хотя по центру или справа – тоже вполне подходящие для него места, если это правильно обыграть.

**Меню.** Что касается меню, то традиционно, если речь не идет об имиджевом сайте с абсолютно нестандартным размещением элементов, оно располагается либо где-то вверху, либо слева. Тем не менее, иногда меню может находиться и с правой стороны: это допустимо при создании второстепенного меню.

SEO Werkz (рис. 69) – это пример дизайна меню современного сайта – чистого и адаптивного, в плоском стиле.



Рисунок 69

Существует еще и такая категория, как промосайты, имиджевые сайты, презентационные и прочие сайты, реализованные с помощью флеш технологии (flash). Здесь, как правило, дизайнер старается отойти от всяких стандартов и придумать собственную навигацию.

Однако какое бы меню вы ни разрабатывали, постарайтесь все-таки сделать так, чтобы оно легко обнаруживалось: было достаточно крупным и находилось примерно в тех областях страницы, где его ожидают видеть.

Раньше было принято вдобавок к меню, традиционно расположенному вверху страницы, делать дублирующее меню в подвальной области. При этом считалось хорошим тоном, когда верхнее меню было реализовано исключительно с помощью графических элементов. Сейчас редко кто делает полностью графическое меню, но, с другой стороны, редкий пользователь отключает в своем браузере отображение графики.

Вы вполне можете сделать внизу дублирующее меню, но на деле сейчас им пользуются немногие, и в такое подвальное меню обычно убирают служебные ссылки: информацию о сайте, о правилах использования информации, связь с администрацией сайта и т. п.

В последние годы в подвале размещают еще и рекламные ссылки, а также SEO-ссылки (*SEO-ссылки – это ссылки, поставленные с целью повлиять на поисковую систему, а не с целью помочь пользователям*) – ссылки на разные разделы сайта по ключевым словам.

**Строка поиска.** На самом деле далеко не каждому сайту или приложению нужна такая функция, как поиск информации по всем его страницам. Нет смысла в поиске на сайте из пяти страниц (что там можно потерять?). А вот если речь идет о достаточно крупном сайте, то поиск очень нужен, особенно если структура его нелогична и запутана.

Учтите: если пользователи слишком часто ищут на вашем сайте, казалось бы, очевидные вещи, значит, при проектировании его структуры были допущены критичные ошибки.

Строку поиска часто гордо помещают в самом верху страницы, буквально рядом с логотипом. Но всегда ли это уместно? На обычном корпоративном сайте вся информация должна легко обнаруживаться и без обращения к функции поиска, поэтому не занимайте нужное место ненужными элементами.

Другое дело – дизайн для интернет-магазина. Если этот магазин действительно огромный и меню его обширно, то для скорости, вероятно, пригодится и поиск. Куда поместить строку поиска в данном случае – решать вам. Можно вверху, сделав на ней акцент, можно где-нибудь справа или слева. Желательно, чтобы поиск обнаруживался на первом или втором экране, но это не догма. Часто строку поиска размещают в самом низу сайта, и этот вариант постепенно становится негласным стандартом для корпоративных сайтов.

И наконец – дизайн сайтов поисковых систем. Присмотритесь к самым популярным из них: строка поиска доминирует над всем остальным, и это правильно. В поисковых системах именно возможность поиска – это то, зачем пришел пользователь, и никакие графические элементы не должны отвлекать его от главного.

**Кнопки.** Этот инструмент, первоначально задуманный для взаимодействия с сайтом, со временем приобрел множество дополнительных функций, о которых можно и нужно рассказать отдельно и подробно. По кнопке можно щелкать, «навешивать» на нее самые разные функции, менять ее дизайн. Главное, что нельзя делать с кнопкой, – это делать из нее ссылку. Нельзя в том смысле, что нехорошо так делать, нецивилизованно, хотя технически реализовать это несложно. Сейчас в Интернете остались редкие сайты-динозавры, использующие кнопки вместо ссылок.

Во-первых, кнопка создана не для этого, а для отправки данных, введенных в форму, а во-вторых, при наведении на кнопку указателя мыши невозможно заранее увидеть в строке состояния браузера, куда же она ведет.

А вот если вы хотите обратить внимание человека на какую-то очень важную ссылку, можно прибегнуть к кнопке. Она может быть достаточно крупной и яркой и размещаться на самом видном месте, но в этом случае такая кнопка должна быть единственной! Двадцать очень заметных кнопок, каждая из которых влечет и манит к себе, – зрелище не для слабонервных.

Как размещать стандартные кнопки? Разумеется, рядом с другими элементами формы. Нарисовали строку поиска – будьте любезны справа от нее поставить кнопочку. Создали блок из полей ввода (например, для отправки почтового сообщения с сайта) – и сюда тоже кнопку. Правило для размещения кнопки рядом с полями ввода и другими элементами форм можно сформулировать примерно так: одно поле ввода – кнопка рядом, два и более полей, заполняемых одновременно, – кнопка внизу, выровненная по левому, правому краю или по центру.

Часто кнопка служит и другим целям. В интернет-магазинах аппетитной кнопкой обычно заменяют ссылку «**Купить!**». При этом часто визуально кнопкой является не что иное, как небольшая картинка, которой присвоена ссылка.

**Баннеры.** Поговорим о самом нелюбимом – о графических баннерах. Если вы делаете сайт, на котором его владелец планирует размещать рекламу, лучше заранее узнать, какую именно и где. Хотя дизайнеру хотелось бы, чтобы ничто не портило его детище, скорее всего, самые дорогие баннерные места будут располагаться высоко и на виду, а вовсе не где-то в подвале. Задача дизайнера – сделать так, чтобы баннер, каким бы он ни был, вписывался в сетку сайта.

Лучше всего встраиваются в дизайн: текстовая контекстная реклама наподобие Google AdSense (AdSense – это простой и бесплат-

ный инструмент, который позволит вам зарабатывать, показывая рекламу на своем сайте.), «тизеры» (реклама с картинкой стандартного размера и текстом), затем – флеш-баннеры, растягивающиеся по заданной ширине, и наконец – реклама одной строкой на всю ширину сайта. Реклама на современном сайте – это неизбежность, которую нужно просто принять как часть нашей повседневной работы.

**Графические элементы.** Нет абсолютно никаких правил относительно того, где должны располагаться графические элементы. Фотографии и иллюстрации могут находиться в шапке сайта, а могут быть равномерно разбросаны по всей странице, тогда как шапка будет отсутствовать как класс. Чаще всего, конечно, самое большое количество изображений располагается вверху, чтобы зрителя с первого взгляда сбивала с ног нереальная красота, оставляющая неизгладимое впечатление. Главное, чтобы за графикой не потерялось все остальное – контент, по которому пользователь поймет, где он находится и о чем все это.

**Элементы главной страницы.** Есть несколько традиционных вещей, которые регулярно размещают на главных страницах сайтов, особенно корпоративных.

**Во-первых,** это Welcome – текст: какое-нибудь приветствие, краткий рассказ о компании, а в запущенном случае – фотография ее директора. Описывать содержание сайта и показывать лицо директора, конечно, вовсе необязательно. Такое небольшое вступление располагают обычно сразу под шапкой сайта.

**Во-вторых,** раздел новостей компании. Блок с новостями присутствует, наверное, на 90% корпоративных сайтов, и чем реже новости обновляются, тем более спорен сам факт присутствия новостного блока. Считается так: если у компании действительно есть о чем писать хотя бы раз в неделю (а лучше раз в два-три дня), то пусть новости будут. Если же что-то новое появляется раз в месяц, то постоянно висящий блок попусту занимает место. В этом случае можно предусмотреть «плавающее» место для новостей где-ни будь вверху: пока новость достаточно свежа, пусть она будет на видном месте в особом блоке. Как только она становится мало актуальной, она исчезает и автоматически переключивается в архив новостей.

**В-третьих,** краткий перечень услуг, предоставляемых компанией, либо три-четыре самых популярных товара, которые она продает или производит. Чем «вкуснее» и интереснее будет подан этот раздел, тем лучше. Это на самом деле самое главное из того, что должен увидеть пользователь, все остальное – во многом лишь дань традиции.



Есть еще самые разнообразные **дополнительные элементы**: голосования, подписка на рассылку, счетчики посещений, календари, блоки с обширной контактной информацией, ссылки на другие сайты по той же тематике.

Как располагать все это, да и стоят ли эти элементы того, чтобы вообще появиться на сайте, – зависит от многих факторов. Добавление любых функций на сайт должно быть осмысленным и практически оправданным.

В целом для главной страницы можно сформулировать такое правило: **только самая ценная и важная информация**. За остальным пользователь пройдет в другие разделы, если сочтет нужным. На то нам и даны ссылки.

**Внутренние страницы.** На внутренних страницах также не должен твориться безудержный бардак: именно ради упорядочения информации и создаются всевозможные разделы и подразделы.

Прорабатывая прототип внутренних страниц, опирайтесь на уже созданное – на главную страницу и на общую сетку. Можно сохранить расположение блоков для всех страниц сайта, что так часто делается на новостных сайтах и крупных порталах. Второй вариант: расположение элементов главной страницы сложнее и прихотливее, блоки на внутренних страницах располагаются более традиционно. И третий вариант: для каждой страницы (или ряда страниц) разрабатывается своя схема расположения контента. Так часто бывает, если на разных страницах располагается разная информация: тексты, фотогалерея, видеофайлы, чат, форум и т.д.

Но в любом случае нужно сохранять единообразие оформления всего сайта. Во-первых, это модульная сетка. На ее пропорциях держится все. Во-вторых, это цветовое и графическое оформление сайта. Логотип, главное меню, подвальная область и шапка (или ее элементы) должны находиться на своем месте и никуда не мигрировать. Какие-то отступления могут быть, но визуально страницы всегда должны быть связаны одна с другой.

## ЛЕКЦИЯ №8. ЮЗАБИЛИТИ-ТЕСТИРОВАНИЕ

Юзабилити-тестирование появилось в крайне бюрократизированных областях – в военной промышленности и сфере рискованного гражданского производства (самолеты, электростанции и т.п.).

**Ю-тестирование** это исследование, выполняемое с целью определения, удобен ли искусственный объект (такой как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения, основанное на привлечении пользователей в качестве тестирующих и суммировании полученных от них выводов.

**Ю-тестированием** является любой эксперимент, направленный на измерение качества интерфейса или же поиск конкретных проблем в нем. Его проводят на всех стадиях создания ПО: планировании и подготовке, активной разработке, приемки и эксплуатации. Тестируется одно и то же, но с разных сторон.

Однако, ошибки, допущенные на стадии проектирования, иногда обнаруживаются, когда уже имеется законченный продукт. Тестирование на этой стадии является наиболее сложным и важным.

Тестирование позволяет:

- Понять, насколько плохо или хорошо работает интерфейс, что может либо побудить улучшить его, либо, если он уже достаточно хорош, успокоиться; в любом случае достигается польза.
- Сравнить качество старого и нового интерфейсов и тем самым дать обоснование изменениям или внедрению.
- Найти и опознать проблематичные фрагменты интерфейса, а при достаточном объеме выборки также и оценить их частотность.

В то же время ю-тестирование не может сделать из плохого продукта продукт хороший. Оно всего лишь делает продукт лучше.

Первое тестирование крупных систем всегда показывает, что интерфейс работает гораздо хуже, чем думает его владелец или создатель, и гораздо лучше, чем изначально уверен тестер.

### **Что и как можно проверить?**

1. Проходимость пользовательских сценариев.

*Задание: выполнить целевое действие.*

2. Востребованность и впечатления от системы

– Сравнение с конкурентами.

– Опрос: что понравилось и будет ли пользоваться?

3. Понятность навигации.

*Задание: найти нужный предмет/функцию в меню/каталоге.*

4. Полноту и доступность контента

***Задание:** найти и объяснить информацию.*

5. Восприятие дизайна.

– Тепловые карты и карты саккад.

**Что можно измерить?**

1. Показатели выполнения заданий.
2. Проблемы, с которыми столкнулись пользователи (частотность, повторяемость, критичность).
3. Время на выполнение задания.
4. Удовлетворенность выполнением задания.
5. Удовлетворенность системой в целом.
6. Эмоциональное восприятие системы или ее частей.
7. Ошибки, которые пользователь допустил.
8. Ожидания пользователя.
9. Глубина просмотра страниц.
10. Достижение целей на конверсионном пути.

**Методы юзабилити-тестирования:**

- «Коридорное» Ю -тестирование
- Модерируемое удаленное тестирование
- Немодерируемое удаленное тестирование
- А/В тестирование

**Коридорное тестирование.** Название коридорное пошло от начальной идеи ловить 5-6 случайных человек, проходящих мимо по коридору. Респондент выполняет задания, которые дает ему модератор, на компьютере или ином устройстве. Часто используется специальное ПО и оборудование для записи сессий. Вариации: тестирование в оборудованной лаборатории (лабораторное тестирование), выездные сессии, вовлеченное наблюдение за пользователем на его рабочем месте.

**Удаленное модерируемое тестирование.** Тестирование проводится один на один, удаленно. Респондент и модератор общаются удаленно (по телефону, скайпу и т. п.). Респондент выполняет задания со своего компьютера, подключившись к совместному удаленному рабочему столу.

**Удаленное немодерируемое тестирование.** Респонденты выполняют тесты, созданные в одной из специальных систем, самостоятельно. Задания для тестирования формируются в одной из систем для проведения удаленного тестирования. Ссылка на тест рассылается респондентам. Респонденты самостоятельно проходят тест.

В системе можно задавать любые вопросы (как в анкете). Выполнение заданий происходит на сайте.

Фиксируемые метрики: выполнение заданий (при настроенных условиях), пути пользователя по сайту, время на выполнение, ответы на вопросы, тепловые карты движения мыши по сайту.

Инструменты с помощью которых можно осуществлять данное тестирование:

- Loop’11 (<https://www.loop11.com>).
- Usabilla (<https://usabilla.com/>).
- UserZoom (<http://www.userzoom.com/>).
- Webnographer (<http://www.webnographer.com/>).
- Еще много (<http://remotereseach/tools/>).

**А/В тестирование** – автоматизированная проверка двух или более версий одного контента на одинаковых аудиториях.

Несколько версий контента (страницы, изображения, письма) с незначительными различиями показываются большим группам пользователей. По результатам статистической выборки измеряются показатели каждой версии (конверсия в целевое действие/время, затраченное на просмотр или поиск информации и т.п.).

На успех теста влияют: понятные различия (можно точно сказать, что повлияло на поведение пользователей), наличие большой репрезентативной выборки респондентов (лучше больше 1000 для каждой версии), однородность выборок для каждой версии.

Инструменты А/В тестирования:

- Content Experiments GA;
- Optimizely.com;
- ABTest.ru.

### **Этапы Ю-тестирования**

Ю-тестирование включает следующие этапы:

1. Определение проблемы.
2. Формирование гипотез.
3. Определение метрик для тестирования.
4. Определение персонажей и сценариев.
5. Подбор респондентов.
6. Заполнение анкеты.
7. Вводный инструктаж.
8. Проведение ю-тестирования.
9. Опрос респондентов.
10. Анализ результатов.
11. Определение требований для проектирования сайта.

**Определение проблемы.** Вначале необходимо сформулировать проблему. Например, «Пользователи заходят на страницу (интерес есть), но не выполняют целевого действия. Почему?»

**Формирование гипотез.** Необходимо выдвинуть свои предположения почему на данной странице возникают проблемы. Например, предполагаем, что пользователи могут: не видеть кнопки/ссылки/ заголовки/разделы, двойственно понимать назначение элементов управления, недостаточно иметь информации, неправильно воспринимать информацию.

**Определение метрик.** Мы должны определить, что будем измерять? В качестве метрик можно взять следующие: справился / не справился, ошибки, отклонение от идеального сценария, понятность сообщений, последовательность действий, время на выполнение задания.

**Определение персонажа.** Персонаж – это собранный образ вашего целевого посетителя. Определить персонажа необходимо для понимания кого необходимо приглашать на тестирование. Нет смысла тестировать бухгалтерскую программу на медработнике, а интерфейс кассового аппарата на художнике.

**Определение сценария теста.** Необходимо составить не больше 10 ситуативных заданий приближенных к реальным сценариям взаимодействия. Сценарий теста – это задание, которое необходимо выполнить пользователю на сайте с предысторией. Например: «Ваша первая учительница организывает встречу выпускников. Она всем разослала ссылку на сервис для выбора удобных дат. Сейчас перед вами откроется сервис, вам необходимо отметить 3-4 даты, когда вы можете прийти на встречу с одноклассниками».

Задания на тестирование не может быть такого вида: «Купите телевизор». Необходимо составить задание приближенное к реальным сценариям взаимодействия. Например, – «У вас сломался старый телевизор. На новый у вас ограниченный бюджет в 500 руб. У вас светлая мебель и обои рядом с телевизором и вы хотите просматривать скачанные на компьютере фильмы на телевизоре. Подберите подходящий вам вариант».

### **Экспертная оценка**

Крупный недостаток Ю-тестирования – высокая стоимость. Более быстрым и дешевым способом проверки качества интерфейса являет-

ся экспертная оценка. Она позволяет обнаружить порядка 80% проблемных мест.

Эксперт (или несколько) проводят аудит системы. Ярче всего достоинства экспертной оценки проявляются в сравнении ее с Ю-тестированием (табл. 4)

Таблица 4

Ю-тестирование	Ю- экспертиза
Нужны пользователи	Нужен эксперт
Объективные данные	Субъективные данные
3 – 4 недели до получения результата	Результат с первого дня
На выходе обозначение проблем	На выходе обозначение проблем + вариант решения

Ю-тестирование находит крупные проблемы интерфейса, но оно не может показать, к примеру, некорректные фрагменты интерфейса, недостаточно серьезные, чтобы вызвать человеческие ошибки, однако снижающие скорость работы пользователя. Опыт же эксперта позволяет выявить их без труда.

Один или несколько экспертов профессиональных дизайнеров интерфейса или Ю -специалист по заданным сценариям или метрикам оценивают удобство системы, фиксируют найденные проблемы и (опционально) дают свои рекомендации.

### Виды экспертной оценки

Чтобы сделать поиск проблем результативным, нужен хоть сколько-нибудь формальный подход. Наиболее распространенные формальные подходы: проверка по контрольному списку, эвристическая оценка, мысленная прогонка по интерфейсу.

**Проверка по контрольному списку.** Проверка по контрольному списку ближе всего к формальному тестированию качества. Составляется список произвольных требований т. н. чек-листы, после чего интерфейс проверяется на соответствие этим требованиям. В контрольном списке может быть все что угодно, как правило, туда прямым ходом отправляются требования из стандартов и руководств по проектированию интерфейсов. Метод, вообще говоря, чрезвычайно надежен при поиске дурацких проблем, вроде грамматических ошибок в ин-



терфейсных тестах. К сожалению, он имеет два принципиальных недостатка:

- Чем более полон и детален контрольный список, тем дольше производится проверка. В хорошем списке может быть больше сотни пунктов. Проверки должны выполняться последовательно, больше чем один пункт за раз проверять нельзя. Предположив, что одна проверка занимает пятнадцать минут, можно посчитать длительность полного исследования.

- Контрольный список, чтобы быть совершенно надежным, не имеет права содержать сколько-нибудь размытых требований.

- К сожалению, достаточно полный контрольный список, содержащий только строго определенные требования, есть явление недоступное.

- Качество проверки по контрольному списку напрямую зависит от качества самого списка. Чтобы создать качественный список требований, нужно затратить годы труда – и самое обидное, что этот список никогда не будет завершен, поскольку все время проявляются новые требования.

С другой стороны, четкий контрольный список может использоваться кем угодно, что дает возможность вынести проверку интерфейса из деятельности Ю-специалиста, передав ее отделу контроля качества.

### **Анализ задач/характеристик**

Представленные методы Ю-тестирования и экспертная оценка довольно хороши. В то же время существует более эффективный метод – анализ задач/характеристик.

Крупной проблемой вышеперечисленных методов оценки является игнорирование самого интересного – пользовательских задач, характеристик целевых пользователей и контекст использования. Кроме того, искать проблемы более продуктивно, если концентрироваться не на самих проблемах, а сразу на способах их решения. Нужно рассматривать интерфейс, задавая себе вопрос «как это улучшить?».

Метод анализа задач/характеристик заключается в следующем:

Сначала составляются: список пользовательских задач, описание контекста использования и краткая характеристика пользователей (например, в формате персонажей). Затем все окна, экраны и пользовательские задачи (т.е. маршруты пользователей по интерфейсу) последовательно рассматриваются под углом следующих вопросов: как здесь ускорить работу данных пользователей в данном контексте, как здесь снизить число ошибок данных пользователей в данном контексте, как

здесь повысить удовлетворенность данных пользователей в данном контексте, как здесь сделать интерфейс самопонятнее данным пользователям, чтобы повысить скорость обучения в данном контексте.

Все предлагаемые изменения, как следствие, находятся на местах каких-либо проблем, каковые и следовало выявить. Что особенно хорошо, сразу же становится ясно, как эти проблемы исправлять.

Суммируя, можно сказать, что экспертная оценка – метод, конечно, неполноценный, но достаточно эффективный и замечательно дополняющий Ю-тестирование. Использовать ее без тестирования разумно только в ситуациях, когда подобрать адекватных респондентов для тестирования по каким-либо причинам невозможно. С другой стороны, «чистое» Ю-тестирование, лишенное параллельной экспертной оценки неэффективно.

Если проводить и тестирование, и оценку, результаты становятся наиболее полными. Тем более, что составление списка пользовательских задач, формализация контекста использования и целевой аудиторией продукта делаются и при тестировании, и при оценке.

Если же сначала провести не особо формальную экспертную оценку, исправить ошибки и протестировать уже обновленный интерфейс, тестирование будет наиболее эффективным.

**Эвристическая оценка.** Эвристическая оценка была разработана Якобом Нильсеном и Рольфом Моличем, которые надеялись с ее помощью сократить продолжительность проведения проверки по контрольному списку. При эвристической оценке вместо десятков и сотен конкретных требований интерфейс проверяется на соответствие всего нескольким общим принципам. Сам Нильсен (Молич впоследствии самоустранился) рекомендовал следующие принципы:

1. В любой момент времени система показывает, что с ней происходит. Этот принцип означает, что пользователь **ВСЕГДА** должен знать, что происходит и на каком участке пути он остановился. Если это сложная регистрация, указывайте, что это 2 шаг из 3. Если что-то закачивается или работает какой-то скрипт – выводите % обработки.

2. Система использует термины, понятия и метафоры, присутствующие в реальном мире, а не обусловленные компьютером.

3. В любой момент пользователь контролирует систему, а не наоборот. Любую команду можно отменить или повторить. Пользователь всегда должен контролировать ситуацию. Например, при заполнении формы обязательно должна быть кнопка «очистить форму». Если форма предусматривает несколько шагов – пользователь должен

вернуться на предыдущий шаг, или наоборот, по возможности – пропустить какой-то, чтоб попозже вернуться к нему.

4. В любой момент времени система выглядит и функционирует единообразным и стандартным способом. Кампания Микрософт в своих продуктах придерживается этого принципа (хотя часто и нарушает многие другие). У них во всех продуктах одни и те же иконки находятся в одних и тех же местах и выглядят одинаково.

5. Интерфейс системы препятствует появлению человеческих ошибок. Ошибку проще предупредить, чем исправить. Везде, где можно упростить выбор и убрать ненужные, случайные действия – следует убрать. Если вам что-то известно о вводимой информации – подсказывайте ее пользователю (формат телефона или код региона) и т.д. В данном случае, пользователь не может пройти дальше, пока не разберется с этими настройками и не выберет хотя бы один вариант анализа сайта. Это намного удобней, чем потом сообщать пользователю что-то типа “выберите хотя бы один вариант, или неверно заданы параметры.)

6. В любой момент времени интерфейс показывает объекты и команды сам, не требуя от пользователя вспоминать их. Максимально упрощайте жизнь пользователю, делайте ему подсказки, запоминайте вводимую ранее им информацию. Например, если у вас опять же многошаговая форма регистрации, показывайте ему уже заполненные поля, если они могут понадобиться в дальнейшем. Кроме того, постарайтесь минимизировать количество текстовых полей, где пользователь должен что-то вводить сам. Давайте ему варианты (подсказки), чтоб он мог выбрать из уже имеющихся вариантов (рис. 70). Подсказка региона выскакивает при вводе хотя бы одной буквы. И если пользователь не знает, как правильно пишется город – он просто кликает по одному из предложенных вариантов.

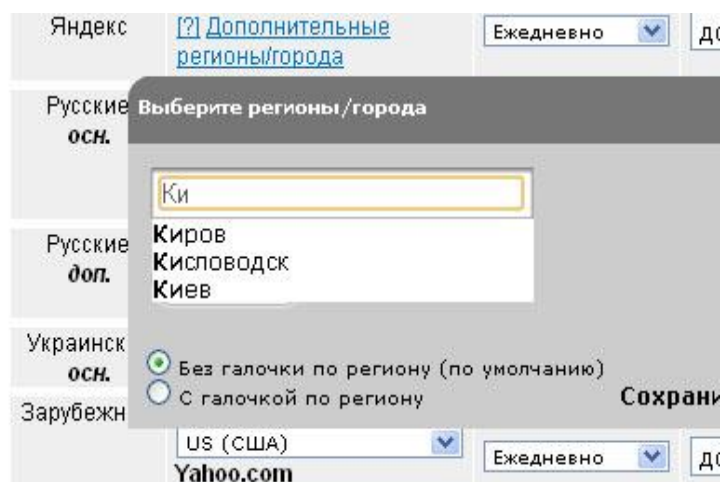


Рисунок 70

7. В интерфейсе есть методы ускорения работы, предназначенные для опытных пользователей и не мешающие пользователям неопытным. Благодаря таким ускорителям опытные пользователи получают резерв для повышения собственной производительности. Одна из самых больших проблем – это как соединить простоту интерфейса и его функционал? Как сделать так, чтоб один и тот же интерфейс был понятен и удобен как профессиональному пользователю, так и новичку?

В Google (рис. 71) мы видим простой поиск для большинства пользователей, а так же расширенный, для продвинутых. При этом функции расширенного поиска спрятаны довольно далеко и только опытный пользователь может их найти, т.е. по сути только тот, кому он и нужен.

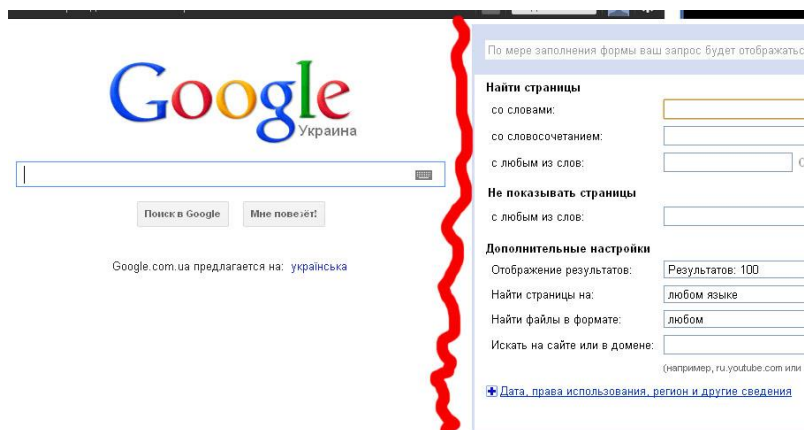


Рисунок 71

8. Интерфейс эстетичен и в любой момент времени не содержит ненужной сейчас информации. К сожалению, этим принципом практически никто не пользуется. В 90% случаев при регистрации вас

спрашивают и домашний телефон, и мобильный, и адрес и e-mail, и даже дату рождения. А все для того – чтоб поздравить вас с новым годом и днем рождения. Все дополнительные данные вы сможете уточнить позже, при необходимости. А изначально, при регистрации например, или при оформлении заказа спрашивайте только ту информацию, которая вам действительно необходима.

9. Интерфейс помогает пользователям обнаруживать и исправлять проблемы, включая человеческие ошибки. Скажите, что означает ошибка базы данных в строке 433? Или длинные жуткие ошибки в строке памяти LXR-XXX-5438645? Все эти системные сообщения должны сохраняться в логах для администратора сайта/системы. Пользователю же пишите нормальным языком, в чем конкретно ошибка. Более того, пользователя по сути интересует только одно: виноват он или система? И можно ли что-то сделать?

Например, при создании почты на рамблере (рис. 72) не только сообщается в чем ошибка, но и предлагаются варианты ее решения. К сожалению, этот принцип на практике применяется крайне редко.

The screenshot shows a web form for creating an email on the Rambler.ru website. At the top, there is a text input field containing the name "Юрий". Below it is another input field containing the word "Маркетинг". The main part of the form consists of two adjacent input fields: the left one contains "marketing" and the right one contains "@rambler.ru". A red rectangular border highlights both of these input fields. To the right of the "@rambler.ru" field, there is a dark grey button with the text "Адрес уже занят" (Address already taken). Below the "marketing" input field, a dropdown menu is open, displaying a list of suggestions under the heading "Свободные адреса" (Free addresses). The suggestions are: "juriyy-marketing@rambler.ru", "juriyy.marketing@rambler.ru", "juriyy\_marketing@rambler.ru", and "juriyymarketing@rambler.ru". To the right of this list, there is a dark grey button with the text "Далее" (Next).

Рисунок 72

10. Справка доступна в любой момент времени. Она достаточна, но не избыточна; к ней легко обращаться; она не абстрактна, а нацелена на решение конкретных задач пользователя; в ней описываются конкретные шаги по решению проблем. Если документ получается достаточно объемным – сделайте по ней краткую навигацию для быстрого перехода по разным разделам, а так же поиск по помощи.

Эти эвристики достаточно результативны. В то же время существуют две проблемы этого метода. Во-первых, оценку должны про-

водить от трех до пяти экспертов, иначе она не будет сколько-нибудь результативной. Требования к числу экспертов постоянно игнорируются. В результате для многих заказчиков, испробовавших этот метод, вся дисциплина юзабилити оказывается полностью дискредитированной. Во-вторых, эвристическая оценка никак не учитывает деятельности пользователей, в лучшем случае она покрывает выполняемые пользователями операции. В результате эвристическая оценка не выявляет структурные проблемы интерфейса, которые, как правило, заметно важнее локальных проблем.

**Мысленная прогонка по интерфейсу.** Если предыдущие методы формализовали правила, которым должен следовать интерфейс, то мысленная прогонка формализует метод, согласно которому интерфейс оценивается. Если исходить из того, что интерфейс предназначен для использования функций, можно проверить, как эти функции вызываются и используются.

Если просто проговорить словами, как работают интерфейсы всех функций, становится понятно, какие из них неоправданно подавлены, а какие работают недостаточно хорошо. Разумеется, для этого тоже необходим опыт эксперта.

Проблема есть и здесь. Хотя метод неплохо работает при простом проговаривании работы функций, наилучшие результаты он дает, если функции не просто проговариваются, а записываются («мысль изреченная есть ложь»). Сам факт записи позволяет увидеть множество противоречий и недостатков в интерфейсе. Увы, запись требует очень много времени, что не позволяет использовать этот метод всегда и всюду.



## СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Купер А. Об интерфейсе. Основы проектирования взаимодействия / А. Купер, Кронин, Р. Рейман – СПб.: Символ-Плюс, 2009. – 649 с.
2. Унгер Р. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия / Р. Унгер, К. Чендлер: Пер. с англ. – СПб. – М.: Символ, 2011. – 327 с.
3. Головач В. Дизайн пользовательского интерфейса. Искусство мыть слона / В. В. Головач – Мн.: БГТУ, 2008. – 94 с.  
[c.http://uibook2.usethics.ru/uibookII.pdf](http://uibook2.usethics.ru/uibookII.pdf)
4. Нильсен Я. Веб-дизайн. Книга Якоба Нильсена / Я. Нильсен. Пер. с англ. – СПб: Символ-Плюс, 2003 – 512 с.: цв. ил.
5. Круг С.. Веб-дизайн: книга Стива Круга, или `не заставляйте меня думать! / Стив Круг. Пер. с англ. – ` СПб.: Символ-Плюс, 2008. – 295 с.
6. Калиновский А. Юзабилити: как сделать сайт удобным / А. И. Калиновский – Мн.: Новое знание, 2005 – 220 с.: ил.
7. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова / Дмитрий Кирсанов, Алина Кирсанова – СПб: Символ-Плюс, 1999 — 376 с.: цв. ил.
8. Гринберг С. UX-дизайн. Идея – эскиз – воплощение / С. Гринберг, Ш. Карпендейл, Н. Маркардт, Б. Бакстон – СПб.: Питер, 2014. – 272 с.: ил.
9. Макнейл П. Веб-дизайн. Идеи, секреты, советы / П. Макейл – СПб.: Питер, 2012. – 272 с.: ил.
10. Купер А. Психбольница в руках пациентов / А. Купер – СПб.: Символ-Плюс, 2004. – 295 с.
11. Бикнер К. Экономичный Web-дизайн / Кэрри Бикнер ; Пер. с англ. Д. С. Ремизова. –М. : НТ Пресс, 2005. – 248 с. : ил. - (Школа Web-мастерства).
12. Уодке К. Информационная архитектура. Чертежи для сайта / К. Уодке – «КУДИЦ-Образ», 2004. – 256 с.: ил.
13. Нейл Т. Мобильная разработка. Галерея шаблонов / Тереза Нейл – СПб.: Питер, 2013. – 208 с.: ил.

### Электронные ресурсы

14. Лунд, М. (2001) Измерение юзабилити с использованием вопросника. STC Юзабилити SIG Информационный бюллетень, 8: [\[Abstract\]](#).
15. Контрольный список Веб-интерфейса  
[http://ddd.exmachina.ru/web/web\\_cheklist/](http://ddd.exmachina.ru/web/web_cheklist/), <http://wiki.software-testing.ru/>

## ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ .....	2
ЛЕКЦИЯ № 1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ .....	4
Пользовательский интерфейс .....	4
Дизайн .....	5
Юзабилити .....	6
Система международных стандартов графических пользовательских интерфейсов .....	9
ЛЕКЦИЯ № 2. ПСИХОФИЗИЧЕСКИЕ ОСОБЕННОСТИ ВОСПРИЯТИЯ ИНФОРМАЦИИ ЧЕЛОВЕКОМ .....	11
Восприятие информации .....	11
Когнетика .....	14
Внимание.....	16
Центральное и периферийное зрение .....	18
Визуальные подсказки .....	20
Память .....	23
Привычки .....	26
ЛЕКЦИЯ № 3. ЭТАПЫ ПРОЕКТИРОВАНИЯ ИНТЕРФЕЙСА .....	27
Проектирование взаимодействия .....	28
Этапы работы над пользовательским интерфейсом.....	33
Сбор функциональных требований (первоначальное проектирование) .....	34
Анализ целей пользователей.....	35
Анализ действий пользователей.....	38
Создание пользовательских сценариев.....	38
Подбор персонажей.....	40
Разработка сценариев.....	42
Проектирование общей структуры.....	43

Выделение независимых блоков .....	44
Определение смысловой связи между блоками.....	45
Создание глоссария.....	47
ЛЕКЦИЯ № 4. ПОСТРОЕНИЕ ПРОТОТИПА. ТЕСТИРОВАНИЕ ПРОТОТИПА .....	48
Первая версия. Бумажная .....	49
Вторая версия – раскадровка .....	50
Третья версия – интерактивный или кликабельный прототип (действующая модель пользовательского интерфейса).....	51
Основные компоновочные блоки макета страницы .....	52
Инструменты для создания прототипов .....	57
Тестирование и модификация прототипа .....	59
ЛЕКЦИЯ № 5. КРИТЕРИИ КАЧЕСТВА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ.....	63
Скорость работы пользователей.....	63
Количество человеческих ошибок .....	67
Уровни ошибок и обратная связь .....	72
Скорость обучения .....	75
Средства обучения .....	75
Субъективное удовлетворение пользователей.....	84
ЛЕКЦИЯ № 6. КОЛИЧЕСТВЕННЫЙ АНАЛИЗ ИНТЕРФЕЙСА .....	96
Расчет по модели GOMS .....	96
Измерение эффективности интерфейса .....	99
ЛЕКЦИЯ №7. ВИЗУАЛЬНАЯ КУЛЬТУРА ДИЗАЙНА ИНТЕРФЕЙСА .....	101
Визуальный дизайн интерфейсов .....	101
Графический дизайн и пользовательские интерфейсы .....	102
Визуальный информационный дизайн .....	102
Строительные блоки визуального дизайна интерфейсов .....	103
Цветовые модели.....	108
Цветовые схемы .....	109

Цветовая гармония .....	114
Шрифтовая схема .....	115
Композиция: вид и ощущение приложения .....	118
Правила композиции.....	119
Размеры экрана .....	127
Сетка .....	129
Преимущества работы с модульной сеткой в дизайне: .....	129
Виды сеток .....	131
Как разместить элементы управления .....	134
ЛЕКЦИЯ №8. ЮЗАБИЛИТИ-ТЕСТИРОВАНИЕ .....	140
Этапы Ю-тестирования .....	142
Экспертная оценка .....	143
Виды экспертной оценки.....	144
Анализ задач/характеристик .....	145

Учебное издание  
**Кишкурно Татьяна Вадимовна**

**ДИЗАЙН И ЮЗАБИЛИТИ  
ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЯ**

Тексты лекций

Редактор  
Компьютерная верстка  
Корректор

Издатель:  
УО «Белорусский государственный технологический университет».  
Свидетельство о государственной регистрации издателя, изготовителя, распро-  
странителя печатных изданий  
№ 1/227 от 20.03.2014.  
Ул. Свердлова, 13а, 220006, г. Минск.